October 2004

SERVO MAGAZINE

Swift Bot Veterans for Freedom

DARPA Grand Challenge: Ladibug on the Prowl

ALSO:

- Genetic Algorithms
- Sound Synthesis
- Secrets to Finding Parts

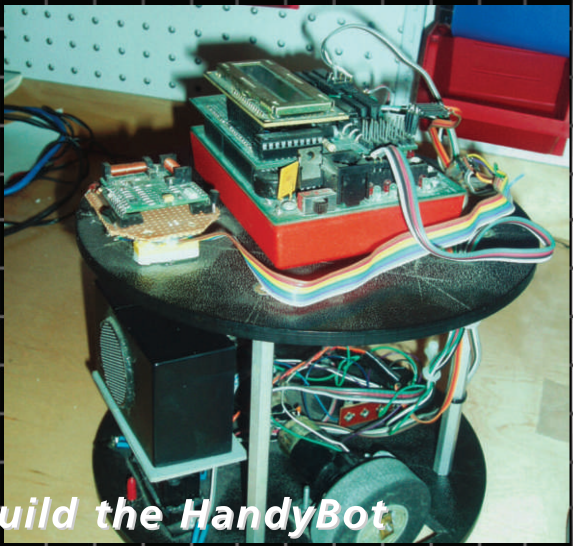
ANYTHING THEY CAN DO...
WE DO...
BASICX24™
$49.95 (QTY 1)
...FASTER
WWW.BASICX.COM
Executing 65,000 lines of Basic code per second the BasicX-24 is the KING of Basic programmable microcontrollers.
400 bytes RAM.
32K User program area.
19 I/O lines with 8 10Bit ADC's.
Real multitasking and Serial UARTs.
...SMALLER
WWW.SITEPLAYER.COM
Siteplayer is a true stand-alone mini web server.
Super easy to use.
Standard RJ-45 network interface.
Control or monitor anything over the web.
SITEPLAYER™
$29.95 (QTY 1)
2X16 SERIALLCD™
$39.95 (QTY 1)
...BETTER
WWW.BASICX.COM
High quality serial 2x16 LCD with backlight
Easy to use.
2400 & 9600 Baud support
Software controllable backlight and contrast.
2x16 Serial LCD
Only $39.95!
Circle #60 on the Reader Service Card.
NetMedia
NETMEDIA INC. 10940 NORTH STALLARD PLACE TUCSON ARIZONA 85737
WWW.NETMEDIA.COM 520. 544. 4567

TABLE OF CONTENTS

Departments

Columns

Take a Sneak Peek!



Build the HandyBot

Coming 11.2004

On the Cover ...

The DARPA Grand Challenge gears up for another round. Are you ready to match the likes of the Ladibug in 2005? (page 9)

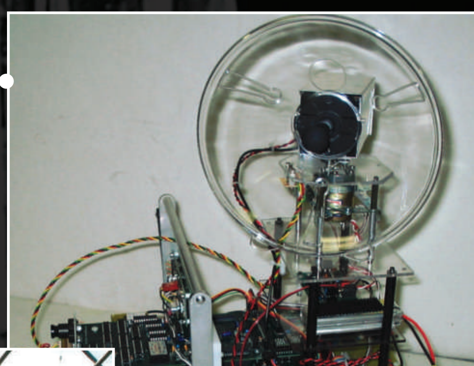
Photo courtesy of Warren Williams.

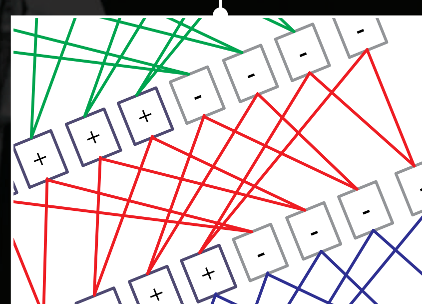
SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. APPLICATION TO MAIL AT PERIODICALS POSTAGE RATE IS PENDING AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, 430 Princeland Court, Corona, CA 92879-1300** or Station A, P.O. Box 54, Windsor ON N9A 6J5. cpcreturns@servomagazine.com

SERVO 10.2004

VOL. 2 NO. 10

Features & Projects





Mind / Iron

by Dan Danknick

As the final month of construction for Tetsujin 2004 looms near, I had the pleasure of spending a few hours speaking with the competitors on the phone to collect data for their profiles (page 58). As the event has matured — both in steel and within their minds — I heard the same comment over and over, "This is awesome because it has so much potential."

Surely, there are many robot competitions in the US today, so what would generate such excitement? Let me share my take on this, based on the ideas in books that, no doubt, you avoided reading back in high school.

In 1905, Upton Sinclair published *The Jungle*, commonly referred to as an exposé of the turn of the century meat packing industry in Chicago. The story centers around Jurgis Rudkus — a young Lithuanian immigrant — trying to get a fair shake for applying his archetypally strong work ethic. At one point, he brags about his physical strength, raising his arms to show his bulging muscles and asks, "Do you want me to believe that, with these arms, people will ever let me starve?" Ultimately, he was proven wrong; his physical strength didn't hold onto its value as much as he'd expected — especially in the face of ruthless business.

Fifty years later, Ayn Rand rolled out her masterpiece, *Atlas Shrugged*, to a post war populace grappling with the role of "big government" overshadowing the decreasing demand for physical labor and the increasing role of thinking people in society.

Her book involves a number of prototypal characters, most notably Hank Rearden, who has alloyed a new form of metal that is stronger than anything else on the market. Industry and the government conspire to keep this "Rearden Metal" from hitting the market and upsetting the status quo that everyone is comfortable with. Unfortunately, this also prevents Rearden from receiving the reward for his mental effort; thus, the plot moves forward as he fights to reap the rewards of his effort.

Tetsujin is unique in that it combines both physical and mental strength, but doesn't rely solely on either. That was the intent from the beginning — not because it is simply unique, but because the robotic challenges of the near future fit this very model. From the outside, it looks like competitors are working on exosuits to lift weights.

In truth, they are working on the technology that fuses the advantages of mechanics to the subtle nuances of the human form. In fact, one competitor commented to me that he was shocked to learn how complex human joints were when he began designing a system to fit around them.In true form, he wasn't discouraged, but excited to apply his mind to the challenge.

When the competitors of Tetsujin raise their power-assisted arms on October 22nd, they will be proclaiming the strength that our culture covets: both the ability to perform work, as well as the intellect to perform it well. I'm tempted to daydream about what people will bring to the competition in 2005. It is, however, the current set of competitors who have stepped up to the plate and will be rewarded for their efforts — an investment in their minds that will never wane. SV

Published Monthly By
The TechTrax Group — A Division Of
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
www.servomagazine.com

Subscription Order ONLY Line
1-800-783-4624

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
robin@servomagazine.com

MANAGING/TECHNICAL EDITOR
Dan Danknick
dan@servomagazine.com

ASSOCIATE EDITOR
Alexandra Lindstrom
alexa@servomagazine.com

CIRCULATION DIRECTOR
Mary Descaro
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
michael@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

STAFF
Janessa Emond
Kristan Rutz

OUR PET ROBOTS
Guido
Mifune

BIO-FEEDBACK

Dear *SERVO*,

I subscribe to your fantastic publication, blah blah blah (Insert standard robot geek babble here.).

Please congratulate whoever is responsible for the great ads found in *Nuts & Volts* and *SERVO*, for example "4 out of 5 mice ..." and RoboSapien surfing. I love these ads and would really like to have poster versions of them.

Geez, maybe I should have searched the site first.

So, if this service doesn't already exist, could you make some crispy, large scale files available for download?

Tob
via Internet

*Editor Dan replies: You can thank Shannon Lemieux in the production department for those great ads! We'll try to have high-res PDF versions of them up on our website (**www.servomagazine.com**) by the time you read this.*

Dear *SERVO*,

My question concerns the June 2004 issue of *SERVO Magazine*, specifically the article "Hack This Budget Airplane for Inexpensive Robot Communication" by L. Paul Verhage. I know the data rate was very low — approximatly one baud — but what was the range?

David Ellis
via Internet

L. Paul Verhage replies:

The box the airplane came in says the range of the radio is 150 feet. However, that wasn't good enough, so I programmed a BoRG Board to flash an LED when it received a signal from the receiver attached to it. After propping it up against a tree, I walked down the street and kept pressing the transmit button.

I was only able to go 200 feet, but — through a pair of binoculars — I could see the LED flash every time I pressed the transmit button. Therefore, the range has to be at least 200 feet in radio quiet environment. I did see some false signals from home electronics or from radio signals bouncing off the walls.

By the way, Wal-Mart is now selling an upgraded version with four channels that they claim has a range of 300 feet. It's only $10.00 more. Ummm, maybe I should look into a research budget.

Dear *SERVO*,

I'd like to comment on September's "Appetizer." First, I must confess that I haven't yet seen *I, Robot*. I'm not into watching movies in theaters. The last time I did, Kirk died.

The biggest turnoff, for me, was the robot riot. We've seen it before; the creation turns on the creator — *Terminator*, *Blade Runner*, HAL, the Golem, *Genesis* Chapter Three, and many others. Been there, seen that — give it a rest!

I'm also surprised that Dr. Pransky did not deduce the meaning of Spooner's rescue: the technical, by being a true Good Samaritan, shames the anti-technical. Perhaps it is Hollywood's "Memo to Self" that technology is not necessarily evil (especially when a supercomputer is used to render the special effects). This is another cliche that needs retiring.

"Appetizer?" I think I'm ready for a mint!

Mike Neary
via Internet

The DARPA Grand Challenge

Live on the Scene — and Behind the Scenes!

by Brian Mork

Living in Southern California has its good points. The week of March 8th, I had a chance to witness trial runs and race day for the DARPA Grand Challenge. Chances are, if you follow the robotic industry, you've already heard of it. It even received coverage on the national news channels.

The overall goal was a race of autonomous robotic vehicles between the area of Barstow, CA and Las Vegas, NV — not remote control vehicles, autonomous vehicles. Once you launch the Autonomous Vehicle (AV), it's on its own for the next 10 hours or so! The course covered about 150 miles of rough sand, gravel, rock, hilly terrain, open plains, steep switch-backs, ravines, road crossings, culverts, ridge-lines, cliffs, and overpasses, sun up to sun down, dust and dirt. It was not a friendly environment.

DARPA Challenge Was — DARPA Challenge Will Be

So, what happened?! Up front, here are the numbers: There were 160 applicants, 86 entries, and 25 teams were invited to the qualifying runs. Fifteen teams qualified for race day under generous terms and four teams made it more than a mile and a half down the course. No one finished the race.

Teams don't give up, though, and DARPA has money left. So ... we'll do it all again in October of 2005! The official web page is at **www.darpa.mil/grandchallenge** A preliminary meet and greet meeting occurred on February 22, 2003 and, for 13 months after that, teams were squirreled away in their shops, turning ideas into reality.

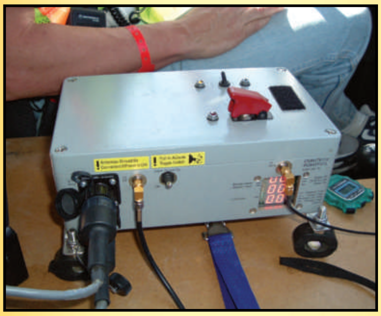
The first public event of the 2004 competition was March 8-12, when the California Speedway in Fontana, CA opened its doors and allowed 25 invited teams to compete in a vehicle Qualification, Inspection, and Demonstration (QID). Score International, which is experienced in setting up Southern California off-road rallies and competitions, was hired by DARPA — the research arm of the Department of Defense — to host the QID. Based on the performance of the teams, they selected competitors for the March 13th main challenge itself. The teams invited to the competition are listed in the Tech Summary, as are results.

Although no team made it past the eight mile mark, the reasons why are a fascinating study of teamwork, funding, resource management, and system engineering.

Budgets for teams ranged from above $5 million of contributed effort to less than $35,000 in winnings from a *Jeopardy* game show. DARPA is especially interested in attracting people who have little or no connection with the government or big defense contractors. That's you! Learn from others' mistakes. Build a better team, recruit a few sponsors, and we'll see you at the races!

It looks terrible! Actually, Team ENSCO's vehicle slowly crawled up the road berm on the left of the road until it couldn't handle the tilt. This was the "easy" stuff in the first mile of the course.

Starting pens: Vehicles headed out of the block toward the audience and then made a shallow left turn as they headed down the first 500 yards of straightaway.



This is the transmit side of the safety remote control system DARPA made every vehicle incorporate. Notice the three position black switch (run — pause — stop) and the red switch — "stop NOW."

Can you identify everything that's hanging on this bike?! The motorcycle team was given the last race slot. After three failed launches, they called it off. It's tricky because tilt/turn control loops have to include knowledge of surface friction. Without *a priori* knowledge, it's difficult to self-tune a control loop in the first 50 milliseconds before the bike falls over.

Navigation Zoom Lens

A successful AV must be able to think at different levels. I sort the navigational tasks as short-, medium-, and long-range tasks. Your AV control programs must be able to quickly zoom in and out, allowing each level to dominate decisions at the appropriate time.

Short Range:

Teams typically used ultrasonic rangers or even physical feelers to determine their short range environment. Challenges representative of the shortest range navigation were built into the qualifying course. On the final stretch, the course designers had added a seductive taper that vehicles couldn't get out of.

Ever see a fish seine net or a lobster trap? How would you navigate a vehicle past a barrier like one, without getting sucked into the dead end? Once you're in the dead end, how would you get out? The logic to do this has nothing to do with what's a mile down the road or even GPS waypoint navigation.

This is a very localized problem that must be solved to get to the waypoint on the other side. A parked car in the middle of the road or narrow farm gates are other examples. A medium-sized bush was the final downfall for the largest vehicle in the competition!

Medium Range:

For any AV to survive, it has to avoid obstacles at the localized level, but it must also respond to line-of-sight observations. The two prevalent technologies used to address this were laser scanners or stereo video systems.

For instance, the berm of the road may be in the direct path from one waypoint to the other and you have to be smart enough to stay on the curving road. This tripped up or turned over more than one vehicle. Things like culverts, gates, overpasses, etc., are all in this medium range category of navigation.

Long Range:

To win a race, an AV has to know where it's going and have some idea of how to get there. The real competition is probably not the other vehicles. It's the 10 hour time limit for the 200 mile course. As one of the engineers observed, "Just getting it to move that fast will be a challenging problem. Maintaining those speeds safely for almost 10 hours is mind-boggling." As a reminder, DARPA is doing the Challenge to service a military need — maybe automatic delivery of weapons, fuel, or food for a soldier battalion, so time does matter.

From your point of view, good speeds tax your vehicle's longest range map planning logic. You could avoid this by simply navigating blindly from waypoint to waypoint. Many "mouse maze" robots ignore a global maze, following simple sensor logic, or build a map memory as it is experienced. Like your commute to work, though, there are probably faster or slower ways to do the trip.

Think of map planning before a family trip. This was Red Team's forte and their vehicle's assertive departure from the staring blocks gave an impression that it knew where it was going. They did go further and faster than any other team.

Safety Systems and Liability

Think of a six-wheeled, 10-ton military vehicle accelerating into a bleacher full of families who have come out to watch the competition — not pretty! This vision is what lead DARPA to take an aggressive stand with respect to safety and liability.

One afternoon, I hung out for a few hours with the chase vehicle crews. A lot of performance choices were left open to the participants. Properly responding to these control boxes was not. It was a required part of passing the QID.

The liability of all these self-controlled robots running around the countryside was covered by an insurance policy purchased by DARPA. Initially, a $300 K limit was announced, but, shortly before race day, they announced an increase to $6 M. It was too late for at least Spirit of Las Vegas; the person doing vision control feared a liability that could have hurt him personally, and had to drop out. I'm sure it's an issue they'll look at for next year.

Mechanical and Control Systems

The competition vehicles showed three basic choices:

1. Modify a commercial, off-the-shelf (OTS) road-worthy vehicle. Most cars and trucks are designed to run 100,000+ miles these days in tremendously varied conditions. These teams leveraged the expertise of auto manufacturers. Typically, they tapped into brakes, gas, gear shifts, and steering. Sensors picked up MPH and engine RPM. You can do welding and

machine work to directly actuate the "people control," but, these days, much of this is electronic anyway and it might be simpler to hook in wherever the traditional control feeds to the onboard computer.

2. Modify a specialty vehicle, such as a golf cart or dune buggy or ATV. To finish the course in the allotted 10 hours, the average speed must be about 20 MPH. There's some pretty rough terrain for a standard vehicle to handle at that speed. Look at the "threat" pictures on the DARPA website. The off-road recreational market makes a number of low slung, stable vehicles to choose from.

3. Build your own frame and build required systems where and how you want them mounted. This is optimized, but requires significant mechanical development. After months of doing this, you're finally ready to start what other teams have already worked on for months.

Control systems varied from a single PC-type computer to racks of specialized CPUs with coordinated system behaviors. Proprietary OSes, Windows, and Linux were used. I noticed that there was a definite trend toward "master computers." In essence, people believed the main issue was the larger navigation goal and the "low level" stuff — such as sensing and actuation — was mentally modularized to be only "inputs" for the power of the "real computer."

No one attacked the problem as a subsumptive collection of simple PLCs (Programmable Logic Computer), which I became very familiar with in the industrial automation world. In fact, the entire PLC and SCADA (Supervisory Control) design paradigm was surprisingly absent.

There were a lot of mechanical buffs and computer buffs. There is a large industry of people who spend their life in between. It would be good to see this design represented. Of course, these experts' liability is that factory plants they design don't have to travel over rough terrain!

Failure Modes

The Tech Summary lists the reasons each team didn't get to the finish line. It mostly lists the "what happened." I think a number of good lessons can be learned by considering the "how" or "why" it happened. A system engineer looks at failures (those that did happen or those that they're trying to prevent) and, often, does a root cause analysis. I'll take you through two examples.

If you do this process for the other teams, you'll be surprised how many issues you think up that need attention when designing a competition vehicle. If you really want to get into the nitty gritty of how other teams have done designs, you can Email me directly and I'll send you technical info from the teams you're interested in.

Team CyberRider's tale was particularly revealing. They did not qualify at the QID, so — obviously — they didn't win the challenge. Intelligent use of the question "Why?" often reveals a chain of failures that could have been stopped at any link. Here's one chain you might follow:

CyberRider didn't win. Why?
Didn't qualify. Why?
Unable to start Qual Run. Why?
Computer failure. Why?
IC failure. Why?
Applied 12 volts to 5 volt chip without spares. Why?
Extreme lack of sleep. Why?
Challenging schedule. Why?
... etc.

In a chain like this, it's a system engineer's job to identify which links are the easy ones to fix, the more "permanent ones," or the least expensive. Two fixes stand out to me immediately. I'm highlighting them specifically because they are not the technical answer that might leap into your mind. They're procedural, and engineers tend to not think this way. You may be the best engineer in the world, but how you ply your trade and what you ask of the people who use your equipment will often have an outcome on projects, programs, and DARPA competitions!

The first thing that stands out is not having a spare. After all the expenses of developing a vehicle, shipping it to the race sites, and air fare and hotels for team members, what's the relative expense of spares? It's probably a good idea to have spares of most every part that is in a critical path!

The second thing is the admitted lack of sleep. Maybe a problem like this can be fixed by having more team members and distributing work loads. Without more bodies, maybe a team needs to designate two people to be well-slept around the clock and always monitor others' work.

I wanted to highlight this vehicle because this root cause is representative of others' troubles, too. For example, the Golem vehicle ran out of power going up a hill. It was because the servo motor on the throttle had been quickly installed during QID when the original burned up and it wasn't calibrated to do more than a light touch on the gas pedal. The problem didn't show up until the vehicle hit a steadily increasing hill.

In the words of Rich Mason, the servo motor didn't get calibrated because of, "the excitement, exhaustion, etcetera." In government test programs, there are mandated crew rest rules and this is precisely the

Warren Williams, who built Ladibug (on the front cover).

TECH

	TEAM NAME	POST MORTEM	START ORDER	MILES	RACE STATUS	HOME	VEHICLE
22	***Red Team***	In switchback, lost sensor, went off course, stradled a berm. Rubber on the front wheels caught fire. Command disabled.	1	7	Disabled	Pittsburg, PA	Red Humvee
21	***SciAutonics II***	Two-thirds of the way up a ridge, vehicle went into an embankment and became stuck. Command disabled.	2	6.7	Disabled	Thousand Oaks, CA	Tomcar Ltd. Dune Buggy
7	***Digital Auto Drive***	Vehicle paused to allow a wrecker to get through, and upon resuming motion, sensors had lost situational awareness. Vehicle hung up on a football-sized rock. Command disabled.	4	6	Disabled	Morgan Hill, CA	Green Toyota Tacoma Pickup Truck
9	***The Golem Group***	Going up a steep hill, vehicle stopped in gear and with engine running, but without enough throttle to climb the hill. Command disabled after 50 minutes.	14	5.2	Disabled	Santa Monica, CA	Black Pickup Truck
5	***Team CalTech***	Vehicle veered off course, went through a fence, tried to come back on the road, but couldn't get through the fence. Command disabled.	3	1.3	Disabled	Pasadena, CA	White Tahoe SUV
20	***Team TerraMax***	Vehicle sensed bushes near the road, backed up and corrected itself several times. At mile 1.2, it was not able to proceed further. Command disabled.	12	1.2	Disabled	Oshkosh, WI	Lime Green Oshkosh 6x6
17	***SciAutonics I***	Vehicle went off the route. Command disabled after sensors tried unsuccessfully for 90 minutes to reacquire the route without any movement.	11	0.75	Disabled	Thousand Oaks, CA	ATV Prowler
4	***Team CIMAR***	Ran into some wire and got totally wrapped up in it. Command disabled.	9	0.45	Disabled	Logan, UT & Gainseville, FL	Tan Custom Frame
13	***Team ENSCO***	Straddled berm, flipped over. Vehicle was removed from the course.	8	0.2	Disabled	Falls Church, VA	White Clamshell ATV
2	***Team CajunBot***	Vehicle brushed a wall on its way out of the chute. Removed from the course.	7	0	Disabled	Lafayette, LA	Gold 6-wheel
25	***Virginia Tech***	Vehicle brakes locked up in the start area. Removed from the course.	5	0	Disabled	Blacksburg, VA	Orange Golf Cart 4x4
23	***Axion Racing***	Sun/shadow spooked. Vehicle circled in the start area. Removed from the course.	6	0	Disabled	Westlake Village, CA	Grand Cherokee SUV
10	***Road Warriors***	Affinity for concrete barriers. Vehicle hit a wall in the start area. Vehicle was removed from the course.	10	0	Disabled	Palos Verdes, CA	Acura SUV
15	***Team Terrahawk***	Problem with the on-board air compressor made the adjustable suspension unworkable.	13	0	Withdrawn	Gardena, CA	Custom Frame
16	***The Blue Team***	Two wheel balance logic failed on sand/gravel.	15	0	Withdrawn	Berkeley, CA	Motorcycle
6	***AL Motorvators***		Non-qualify	n/a	n/a	Los Angeles, CA	Dune Buggy
11	***Team CyberRider***	Application of 12 V burned 5 V IC — lack of sleep!	Non-qualify	n/a	n/a	Irvine, CA	Dune Buggy
18	***Rover Systems***		Non-qualify	n/a	n/a	Santa Ana, CA	Custom Frame ATV
19	***Team LoGHIQ***		Non-qualify	n/a	n/a	Walden, NY	Custom Frame
14	***Spirit of Las Vegas***	Liability threat deterred video system developer. (Notice the team acronym is "Team SOL"; sorry Kent!)	Static display	n/a	n/a	Edwards AFB, CA	Honda ATV
1	***Team Phantasm***	PC board failure.	Static display	n/a	n/a	Ballwin, MO	Kawasaki ATV
12	***Rob Meyer Prod***		No QID	n/a	n/a	Tucson, AZ	
24	***Team Overbot***		No QID	n/a	n/a	Redwood City, CA	
3	***Arctic Tortoise***		No QID	n/a	n/a	Fairbanks, AK	
8	***Incite Racing***		No QID	n/a	n/a	Cary, NC	

SUMMARY

NOTABLE FEATURE	LEADER	LEADER EMAIL	WEBPAGE
Three-axis gimbal for scanners	Red Whittaker	red@cmu.edu	www.redteamracing.org/
	Paul Gunthner	pgunthner@sciautonics.com	www.sciautonics.com/DGC_Elbit/HTML/
No LRF — Only stereo vision	David Hall	dhall@velodyne.com	www.digitalautodrive.com/
$39,000.00 — *Jeopardy* game show winnings	Richard Mason	mason@robotics.caltech.edu	www.golemgroup.com/
	David van Gogh	dvangogh@caltech.edu	www.roversystems.com/
Huge vehicle 24,000 lb. GW	Jim Fravert	jfravert@oshtruck.com ozguner.1@osu.edu	www.oshkoshtruck.com/darpa
	John Porter	jporter@sciautonics.com	www.sciautonics.com/DGC_SciAutonics/
Rotating LRF	Carl Crane	dga@cimar.mae.ufl.edu ccrane@ufl.edu	http://cimar.mae.ufl.edu/grand_challenge/
	Gary Carr	carr.gary@ensco.com	www.ensco.com/news/darpa/index.htm
Self-righting lateral actuators	Charles Cavanaugh	cdc@cacs.louisiana.edu	www.cacs.louisiana.edu/~arun/cajunbot/
	Charles Reinholtz	creinhol@vt.edu	www.me.vt.edu/grandchallenge/
	Bill Kehaly	BKehaly@AxionRacing.com	www.axionracing.com/
High school team	Chris Bowles	bowles@mail.pvpusd.k12.ca.us	www.pvrw.com/
Body articulated steering	Todd Mendenhall	TerraHawk2004@yahoo.com	None published
Two wheel stability study	Anthony Levandowski	anthony@ieor.berkeley.edu	www.ghostriderrobot.com/
	CJ Pedersen	chris@cjpedersen.com	http://autointelligent.com/
Environmentally friendly	Ivar Schoenmeyr	schoenmeyr@aol.com	www.cyberrider.org/
Low CG	Ted Copperthite	ted@roversystems.com	www.roversystems.com/
Electric drive wheels	Seth Cabe	cabes@alum.rpi.edu	www.cabecomposites.com/team.html
	Kent Tiffany	kent.tiffany@netzero.net	www.af.mil/news/story_print.asp?storyID=123007202
Treads	Warren Williams	sparten1@swbell.net bill@zimmerly.com	www.phantasm1.com/welcome.html
Large ground clearance 58" tires	Robert Meyer	rmeyer@robmeyer.productions.com	www.robmeyerproductions.com/
	John Nagle	info@overbot.com	www.overbot.com/
	Richard Ruhkick	arctic-tortoise@gi.alaska.edu	www.gi.alaska.edu/DGC/
	Grayson Randall	grayson@insightracing.org	www.insightracing.org/

Team Terrahawk didn't make it to the main race due to a pneumatic failure in the suspension system.

reason why. Don't underestimate your people needs when trying to win a technical challenge!

I've highlighted two procedural or process fixes. The problem with procedures are: First, they need to be taught to new team members and, second, people may be overwhelmed and the procedures forgotten when other critical events down the road start happening. The military is steeped in procedures, but it uses LOTS of training to make sure soldiers know how to use their equipment, even in bad situations. Without such a time and training luxury, though, it might be better to turn to other answers.

Engineering answers are usually more permanent. In this case, a technical answer can be found and one of the team members already knew it. Next time around, the wires that got hooked up wrong are going to have plugs that are consistently and physically different for 5 V or 12 V.

A more obscure root cause analysis can be seen with the DAD truck. It's a late-model pickup truck — still legal for public road use, but heavily augmented by a sensor and control suite. The chase team commanded it to stop as a race wrecker went by to get another vehicle. When the "pause" command was released back to "run" mode, the vehicle hung up on a football-sized rock and did not go any further.

Running down the root cause chain might yield this:

Didn't win. Why?
Couldn't move. Why?
Couldn't climb over rock. Why?

I stopped the questioning here because, at this stage, there are multiple possible answers and I don't know enough to answer the question. This is typical and often dictates instrumentation efforts to log and document what's happening on a vehicle. Consider: Rock was too big. Tires were too small. Friction of tires was insufficient. Engine was not powerful enough. Momentum stopped by command box. Transmission slipped. Other possible causes were that, once the sensors stopped moving, they lost situational awareness and the "start moving" throttle logic didn't handle impediments. Each can take you down a different path, yielding a spreading branch of possible causes that have to addressed.

This vehicle — TerraMax — basically flattened the berms when it hit them. I'm not sure the concrete blocks or the audience bleachers would have stopped it, either. It made it for just over a mile and then got spooked by a bush. After about an hour of being passed by other vehicles, the team threw in the towel.

When I saw this, two realizations hit me. First, all the fancy computer logic onboard this vehicle may have been good enough to finish the race if it was on a different platform. (We'll never know.) I sort of laughed when I saw the monstrous vehicles by the Red Team, TerraMax, and Terrahawk, but now acquisition of a monstrous vehicle would be a consideration if I had to design something that was going to win rather than just exhibit new engineering.

Your Connection to the Government

I've subscribed to *SERVO*'s sister publication, *Nuts & Volts*, for 12 years. What started as a one-way subscription to learn slowly became a skill, a vocation, and a profession. This may happen to you, too. During my journey, here's a realization I've had: There are lots of engineers in our nation's industry base — even tremendously smart ones — contributing to projects.

However, if you ever want to really have your ideas funded, you need to become a full-time government employee or start a business of your own. Those are the two players that control technical development in our nation. Understanding how the government system works may help you accomplish your goals.

An entrepreneur — Christopher Beskar, who is working to break into government aerospace contracting — has highlighted on his website (**www.stavatti.com**) the difference between selling what's made and making what's sold. I've adapted the concept to the test and acquisitions world I live in by using the comparison, "building to spec or spec'ing to build."

In the acquisition world, the "way it used to be done" is that companies would research and build airplanes, vehicles, or weapons systems to government specifications and the government would test possible designs, pick one, and purchase them from the manufacturer. They were built to spec.

With such tight government specs, we ended up with famous $500.00 toilet seats and such because pre-written

specs didn't accommodate reasonable changes due to new technology or just good alternate ideas by a contractor.

In order to accommodate this, the government started asking for contractors to bid on the system they thought would best answer a need. Of course, awards tend to be given to people who promise a lot for a low price. The trend drifts toward contracts that drag on and on in a cat and mouse game of contractors milking a few more months of income while the government is afraid of cutting a contract because the other possible companies have moved onto other commitments.

If you don't see the DARPA Grand Challenge in that description, you're missing something. With very few limits, DARPA doesn't care how you get to the finish line. Simply demonstrating that you have the capability is accepted as prima facia evidence of the system you designed.

Prize-based industrial funding seems to be the new idea that fits well in this paradigm. There are no direct government dollars during the development cycle, but there's a big payoff at the end. There are no subsystem level government input or parts specs. It's a tacit admission that the contractors probably have more experienced people than the program officers in the military and come up with better answers on their own.

It's happening in other areas, too. Perhaps you've heard of the Spaceship One launch done by Scaled Composites. They were motivated to put the first private pilot into space the same way. The team is looking at a $10 million prize if they're the first to meet the X-Prize criterion. They've spent more than that developing the system, but the prize was enough of a catalyst . The day after their record breaking flight, NASA announced they would look at competitive "prize" motivations to spur innovation.

Big prize money fits our societal predilection for lottery tickets and news reporting that makes statistically rare events seem like they're happening all around us. I'm sure we'll find the flaws in this system later, but — for now — many technology dollars are going this way. As long as it's happening, you might as well be part of it!

... ESTs

It's always fun to find the biggest, fastest, smallest, etc. I call these the "... ests," ESTs, or Extra Special Teams. Here are my picks:

Biggest — TerraMax, by a long shot. You could see the bleacher audience recoil as it came out of the starting pen.

Most Mature Sponsorship and PR — Red Team. This wasn't accidental. It's socially acceptable for companies to donate to educational institutions and this team has it down to an art.

Youngest Team Members — Palos Verdes High School. The energy was palpable. It's cool to watch the mentorship happen.

Most Original — A recent Master's Degree graduate put together a team of 19 college students to balance and propel a motorcycle down the dusty trail. Remember the three zoom levels I discussed earlier? These guys had an even more fundamental task of even getting their vehicle to stay upright! I wonder if they'll be back next year or whether Challenge #1 propelled them onto other high-paying jobs.

Minimal Resources — Team Phantasm. They had a computer failure during QID. It was a PC with the case open. They were scrounging other teams for a spare computer to meet their trial time. It didn't happen and they withdrew.

Minimal budget — This category appeared to be a tie between The Golem Group (Jeopardy winnings) and Spirit of LV (I know what military officers make.).

As the sun was coming up over the horizon, all the teams had been awake for hours (or had never slept). Improvised work stations, networks, camping stoves, and waffle mix all got table space.

Conclusion

In the end, there were a lot of motives for participating in the Challenge and a lot of issues to be challenged by. Why the teams showed up didn't really matter. Some were there for PR. Some were there as part of college education. Some were there to exhibit new designs. Some were there to solicit business. Some were there to improve their skills.

Teams had different personalities. It's an issue of style and preference. If you want a laid back team, it's available. If you want a high-pressure cooker environment, it's available. Develop skills? Push to win? Enjoy a project with friends?

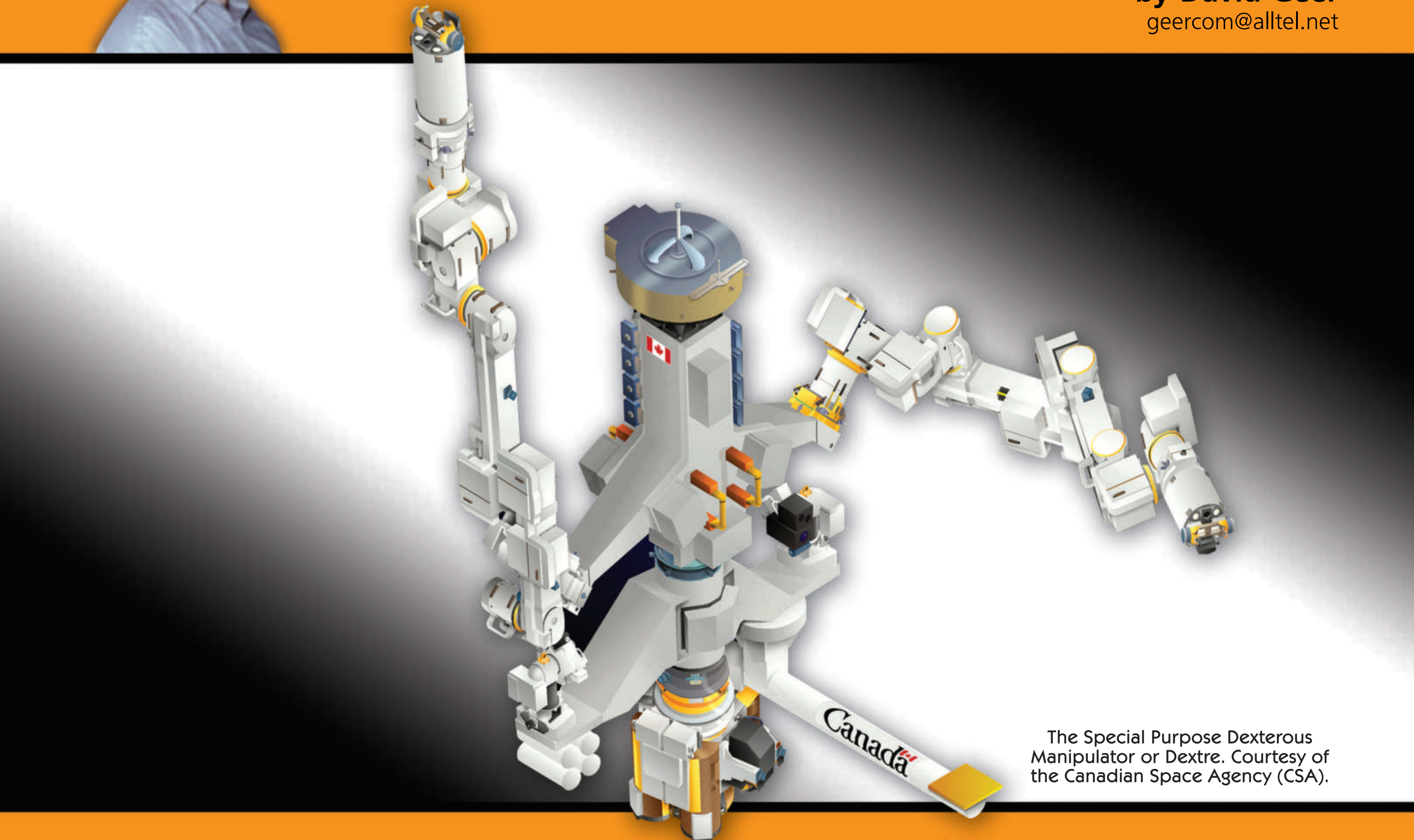
Each in its own way, all teams are contributing to what DARPA needs and, for the Fall 2005 Challenge, who crosses the finish line in the required time is up to any team! SV

About the Author

Brian Mork, Ph.D, is an engineer, scientist, and aviator. He's now serving as a Systems Engineer for Directed Energy Weapons under the Electronic Warfare directorate at Edwards Air Force Base. Contact him through his website at **www.increa.com**

GEER HEAD

by David Geer
geercom@alltel.net

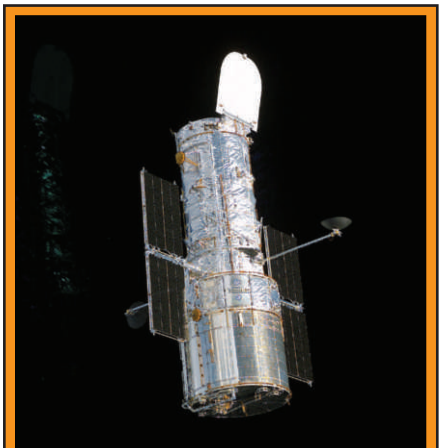


The Special Purpose Dexterous Manipulator or Dextre. Courtesy of the Canadian Space Agency (CSA).

Hubble Trouble? Call Dextre on the Double!

There is seldom a more humbling way to discover someone else's strengths than when you need a helping hand — or a robotic arm for that matter. The Hubble Space Telescope needs a hand now and it may come from a high-flying mechanic north of the border — a robotic Canuck named Dextre.

The Hubble Space Telescope. Photo courtesy of NASA.



Experimental Robonauts in action. Photo courtesy of NASA.

The Hubble was going to be repaired by astronauts who would make their way to it on a special Space Shuttle mission. The mission was cancelled because safety rules that resulted from the last Shuttle accident prohibit the Shuttle from travelling to the telescope.

Background

Canada's contribution to the International Space Station (ISS) is one multifaceted component, consisting of three smaller, yet huge components. The overall beast is the Mobile Servicing System (MSS).

The three subcomponents include the Canadarm2, a.k.a., the Space

Station Remote Manipulator System (SSRMS). An extensive upgrade — for lack of phrasing that would do it justice — to the original Canadarm, the Canadarm2 is much more than the first-gen robotic arm that sat mounted near the Space Shuttle's payload bay doors.

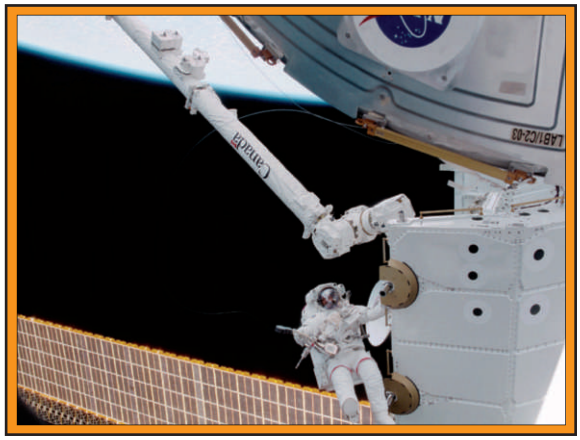
The Canadarm2 has several hands, all capable of anchoring at various locations on the ISS. The Canadarm2 can actually crawl all over the ISS to do its work anywhere it needs to.

The Mobile Remote Servicer Base System (MBS) is the component that will provide a foundation for Canadarm2 and run the length of the ISS on a track, something like a railroad car or roller coaster ride. The MBS has been described as a small truck, which would make the Canadarm2 comparable to a telephone repairperson's lift, with Dextre the repairperson.

The Star Attraction

Dextre — the Special Purpose Dexterous Manipulator (SPDM) robot — is a maintenance expert created for upkeep and repair jobs on the ISS's exterior.

Dextre's job will mostly be to pull and replace small parts on the outside of the ISS, where its "dexterity" will come in "handy." The two-armed robot frankly looks much like a techie mechanic. Its Power Data Grapple Fixture up top is the near perfect, yet (we assume) coincidental, replica of a geek's beanie cap — complete with propeller — worn backwards so Dextre can "see what he's doing" and look cool doing it, too.



The Canadarm2, to which Dextre will attach itself for work on the International Space Station. Photo courtesy of NASA.

Dextre has two shoulders, two arms, and seven joints per arm. Each joint has a tool changeout mechanism

Parts Is Parts

From top to bottom, Dextre includes a Power Data Grapple Fixture for other equipment to latch on to, an upper body, and two arms of seven joints each. The hands at the ends of these arms come equipped with a Force Movement Sensor to help make Dextre careful with his equipment and Orbital Replacement Units (ORUs), which are comparable to what we non-robotic repair technicians here on Earth might call Field Replaceable Units (FRUs). Each hand also has a Tool Changeout Mechanism to secure tools for the job "at hand."

Below Dextre's Electronics Platform, you will find a camera and a light-tilting unit that enable the engineers operating Dextre to see exactly what it is doing. A body roll joint allows Dextre to turn and pivot at the waist. An ORU temporary platform gives it a workbench of sorts. Like every good repairman, Dextre needs a tool belt and its four-tool holder does the trick — actually appearing about waist high on Dextre. Finally, a Latching End Effector enables Dextre to connect securely to the end of the Canadarm2.

The Hubble Space Telescope over Cuba. Photo courtesy of NASA.

that can support various devices, like sockets and wrenches. Dextre has a pivoting upper body and supports itself using one arm to grapple while it works with the other. Dextre will be attached to one end of the Canadarm2 as a hand of sorts. This hand has two arms of its own and the ability to touch and sense with an accuracy comparable to your hand or mine.

Dextre goes to work prepared with flashlight (lighting), eyes wide open (a video camera), and a tool belt (tool holders). Dextre will replace or install computers, batteries, and power supplies, among many other tasks. The overall MSS component was built by MD Robotics, a Canadian firm, on behalf of the Canadian Space Agency (CSA).

The Mission

The Hubble Space Telescope uses an Imaging Spectrograph (STIS) built by Ball Aerospace to take space pics and then examine the photographed space bodies.

The STIS is broken and there are many other maintenance and upkeep issues that need attending to on the Hubble, as well; otherwise, the Hubble will become another useless piece of floating space junk. Though Dextre won't complete testing and be ready for launch until 2005, NASA is willing to wait. Concerns over needless risks and possible repeats of the Columbia Space Shuttle disaster have driven NASA to be reluctant to send a human solution to the Hubble's current problem when a robotic one may eventually be made available.

There is growing support from scientists, astronomers, and other constituents for finding a solution for

NASA's Own Robonaut Is in the Works, but Not Ready

A robotic alternative to Dextre is being considered to save the Hubble; it is NASA and DARPA's joint project, the NASA Robonaut Program. The Robonaut is eventually intended to be a humanoid astronaut replacement for spacewalks.

The Robonaut is generations ahead of its time in space robotics, disallowing the need for any grappling and specialized robot tools. In fact, the goal of the Robonaut Program is to perfect a humanoid robot whose manual dexterity is as good or better than that of a human astronaut in his or her space suit and gloves.

Because much of space hardware is designed to be serviced by people, rather than traditional robots, the robots used to replace people must do the work the same way as people would, with the same tools and manipulations. The Robonaut will eventually be capable of the same range of motion, strength, and duration of work as astronauts on a spacewalk. The Robonauts will be controlled remotely by real astronauts.

Mechatronics is the technology behind the dexterous hands of the Robonauts. The arms are constructed with embedded avionics technology. All input data from sensors on the Robonauts will go to a core data management system, much like a human central nervous system. The robot will even learn from its environment in order to optimize its performance.

Hubble reparations. A robotic solution seems the most sound. In any case, NASA won't be sending Dextre or anything else up for at least three years.

How You Gonna Do It, Dex?

Dextre is compatible for opening the Hubble doors, using the tools necessary to install the needed parts, and replacing batteries and gyroscopes and cameras. It is hoped that Dextre could repair the Spectrograph, as well. Upgrades and repairs are expected to keep the Hubble in good working order for five more years. It needs to be attended to by late 2007, when its batteries run out.

Plans include sending the robot up with a propulsion unit that would attach to the Hubble so that, when it eventually re-enters the atmosphere, it can be guided over unpopulated areas before it breaks up, saving our fellow Earth inhabitants from harm.

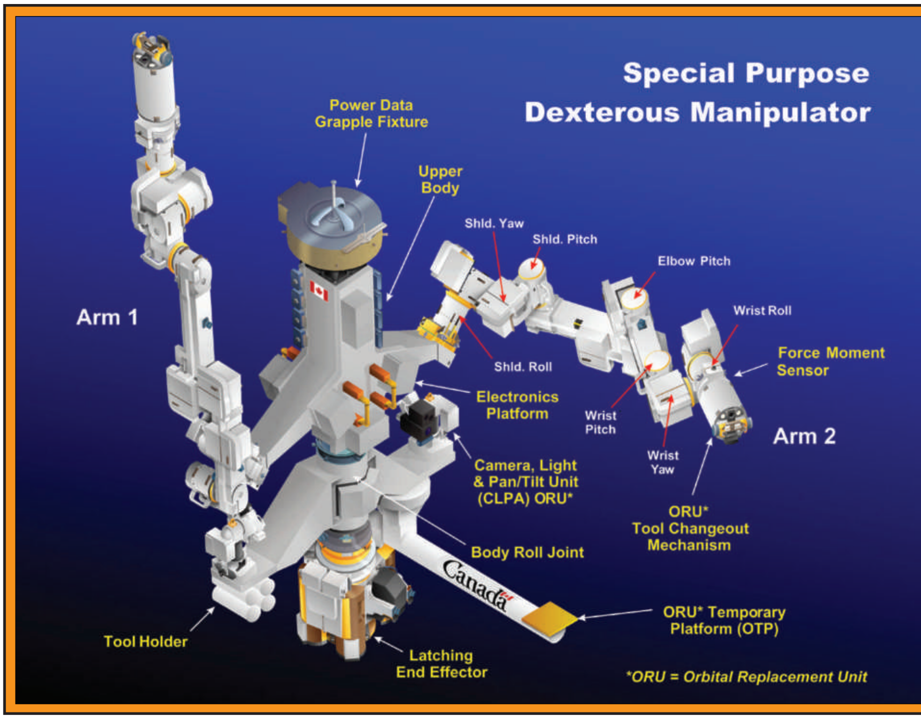
The robot needs to be able to get to the Hubble, attach itself to it using existing grappling, and then complete all the repair jobs successfully. One technology that could help Dextre attach to the Hubble instead of smashing into it is a laser-based radar.

Grappling equipment on the Hubble is compatible with the Canadarm2, which has been used in previous Hubble repair missions. Dextre's grappling equipment would also be compatible.

Two instruments that NASA would be replacing on the Hubble are currently housed in Orbital Replacement Units (ORUs). This is the same type of container Dextre has been built to use to replace parts on the ISS.

The most difficult part of the replacement tasks is the precision required to put the ORUs in place with exactly the same alignment as used to remove the previous units. Otherwise, the ORU could get jammed.

Dextre's memory can record its exact positioning throughout any mission. It can duplicate its positioning and movements exactly to replace the part or instrument. Dextre's sense of touch enables it to sense how much force it is using to replace the units. It would be aware of the slightest resistance if a jam were starting to occur and could back the unit out to try again. After all the hoopla, it is also possible that a robot like Dextre could be built for the job, rather than using Dextre itself. **SV**



Dextre — complete with labels for all his parts.
Photo courtesy of the Canadian Space Agency (CSA).

Resources

Check out the Canadian Space Agency:
www.space.gc.ca/asc/eng/default.asp

Learn more about NASA space flight:
http://spaceflight.nasa.gov

Rubberbands and Bailing Wire

by Jack Buffington

Sounds Like a Winner!

It is often important to have an idea about just what, exactly, is going on inside of your robot. Microprocessors are not very expressive devices unless you give them a way to communicate with the external world. While hooking them up to a serial port or lighting LEDs will give you an accurate idea of what the processor is doing, neither has the simple appeal that having your robot make sounds does. Imagine if you could have your robot make a tire screech and crash sound when it bumps into something. What if it could play polyphonic tones to indicate that it thinks it has completed a task? This month's column will show you how to generate those sounds.

Let's start out with the simplest form of sound generation. This is where you are simply outputting a square wave at the frequency that you desire. The BASIC Stamp's freqout command does exactly this. Figure 1 shows the waveform for two different frequencies using this method of generating sound. That is pretty easy. Let's look at how to connect to something that can turn this waveform into an audible sound. For this type of waveform, you can connect to a piezoelectric disk. These are flat, brass disks that have a small layer of a ceramic material on one side. Most digital watches have one inside. Piezoelectric disks are not good at reproducing many types of sounds, but — for waveforms that are essentially square waves of one amplitude — they do a good job. Keep in mind that piezoelectric disks are not as good at reproducing low frequencies as they are for higher ones.

Another option is to connect to a speaker that has a voice coil and some sort of paper or plastic cone, such as one you would find in a radio. Figure 3 shows how you would connect to a speaker. The capacitor is in there so that you do not burn out the I/O pin of your microprocessor.

Square wave sounds are easy to implement and cost-effective if you are producing a product, but are limited in the expressiveness of the sounds that can be generated. You can improve the quality of your sound drastically by using something called Pulse Width Modulation (PWM). With PWM, you can generate waveforms that approximate any analog waveform.

Figure 1. Square waves at different frequencies.

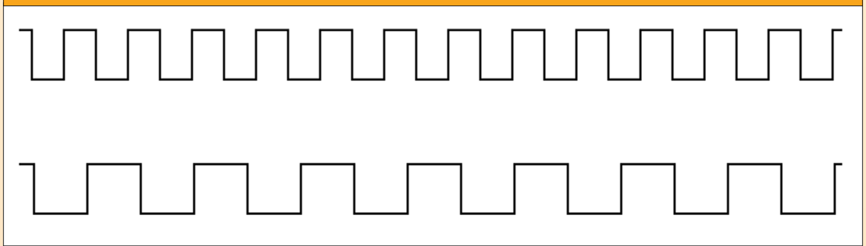


Figure 2. Connecting to a piezoelectric disk.

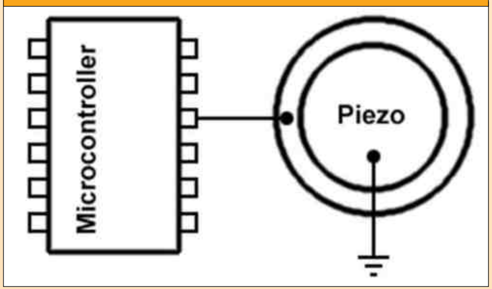
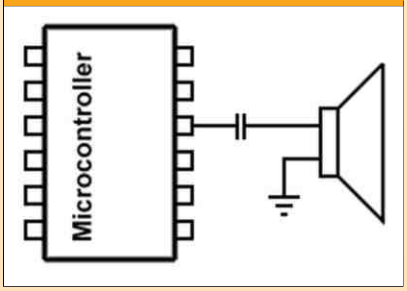


Figure 3. Connecting to a speaker.



Take a look at Figure 4 to see an example of PWM simulating an analog sine wave. For PWM signals, the pulses go high at a regular interval, but go low at *varying* intervals. Notice how the high pulses in the PWM signal are narrow when the sine wave is low and — when the sine wave is high — the high pulses in the PWM signal are wide. If the frequency that the PWM is happening at is sufficiently fast, the speaker will not be able to move fast enough to replicate the square waves that are being put into it, but will instead move in proportion to the width of the PWM pulses.

Figure 4 shows PWM being done in a single-ended manner. Another possibility would be to do it in a differential manner, where you can do pulse width modulation on either of the signal lines going to your speaker or amplifier (Figure 5). This is an improvement over

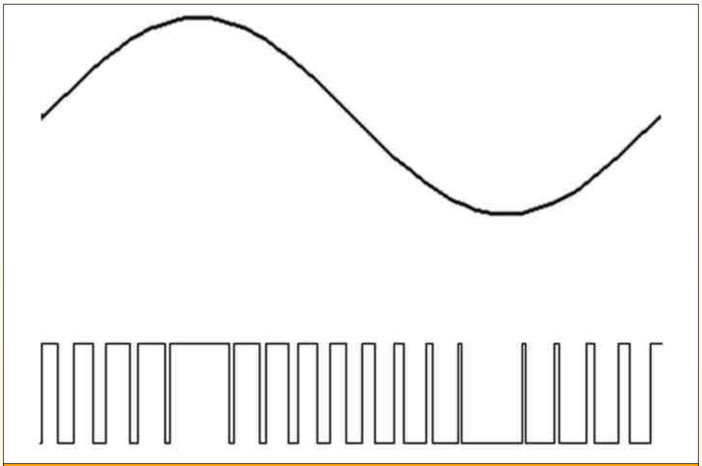


Figure 4. A sine wave and the PWM signal that approximates it.

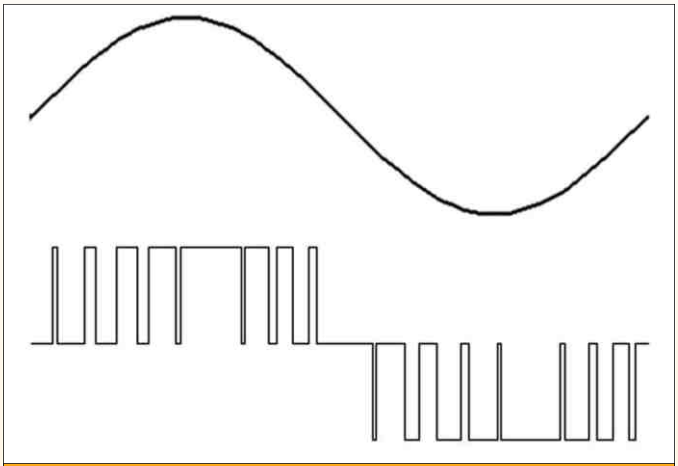


Figure 5. Differential PWM.

single-ended PWM in some situations.

A final option for generating sound using a microcontroller is to use a setup called a resistor ladder. There are two types of resistor ladder setups. One allows you to vary the voltage at its output. The other setup allows you to vary the current at the output. These strategies are shown in Figures 7 and 8. For the variable current resistor ladder, the most significant bit will be your base value in Ohms. Each less significant bit will be half the Ohms of the previous one. For the variable voltage ladder, you will just need two values of resistors. One value will be the base value and the other will be half of that value. The basic strategy for controlling both of these is that you will output a binary number onto the pins connected to the resistors. The resulting voltage or current will be proportional to the binary number.

For all of the strategies discussed so far, you may need some sort of amplification. The resistor ladder methods will certainly need amplification. RadioShack sells some books by Forrest Mims that can get you started with amplification.

Now that the different strategies have been discussed, let's delve a little deeper and talk about how to implement them to get some pleasing sounds coming out of your next robotic project.

Let's first discuss PWM. It may seem difficult to implement, but it actually isn't — with certain exceptions. If you are using a processor — such as the BASIC Stamp — then you are really going to be limited to using the **freqout** command to generate sound. The BASIC Stamp just doesn't have the horsepower to deal with PWM.

Parallax does sell a product called the PWMPAL that will give you access to PWM. Certainly, the easiest way to do your pulse width modulation is to use a microcontroller or other chip that has a built-in PWM module. Using a PWM peripheral frees up your processor to do other things. Just send the PWM value that you desire and forget about it. Most microcontroller families — such as the PIC, AVR, or 8051 — have chips with PWM peripherals. Of course, that isn't always an option because of the processor chosen.

Let's look at what these peripherals are doing internally to understand how to write a program that can do PWM. A PWM peripheral will contain a counter, your PWM values, and a magnitude comparator. The counter will increment on a regular basis. After each increment, it will compare the counter's value to your PWM value. If the counter is less than or equal to your PWM value, then it will output a high value; otherwise, the output will be low.

You may be a bit confused at this point about the operation of the counter. If it is always incrementing, then

Figure 6. Connecting to a speaker using differential PWM.

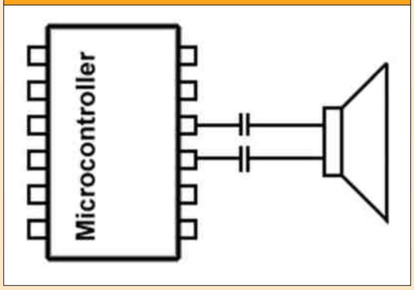
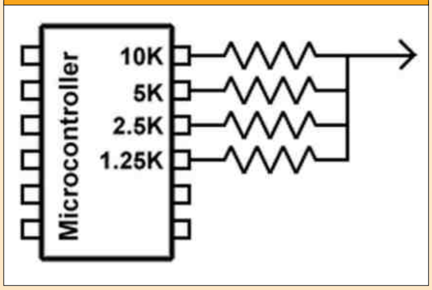


Figure 7. Resistor ladder that outputs a variable current.



TECH TIDBIT

An interrupt routine is a piece of code that is not part of the main loop in a program. Instead, it is run when some sort of event triggers an interrupt. Interrupts can be generated by things such as a timer overflowing, an I/O pin changing, data coming in from the serial port, or many other reasons. When an interrupt happens, the processor will stop executing the main loop and store a pointer to where in the program it was and then execution will jump to the interrupt routine. When it is finished with the interrupt routine, it will look at the pointer and go back to the same place in the program to resume execution.

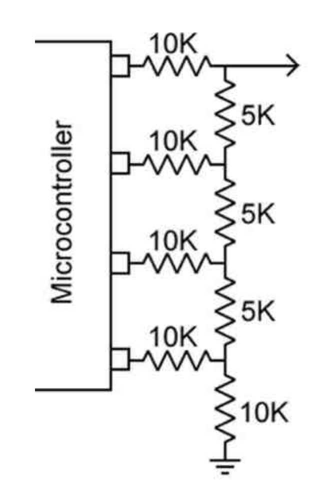


Figure 8. Resistor ladder that outputs a variable voltage.

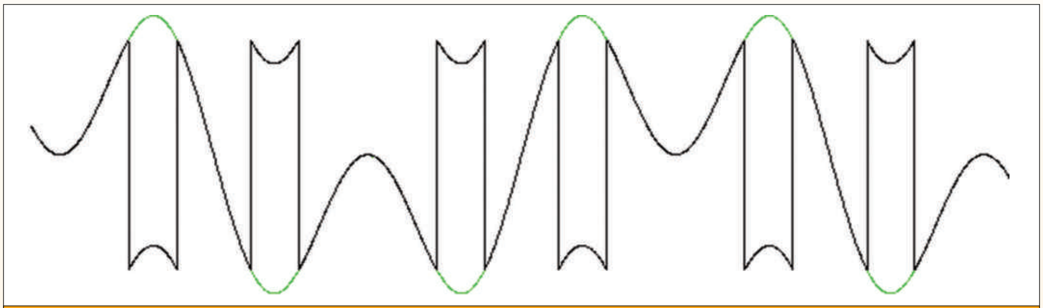


Figure 10. Waveform that results from overflow errors.

```
PWMcount++;
if (PWMcount > PWMvalue)
 output_low(PWMpin);
else
 output_high(PWMpin);
```

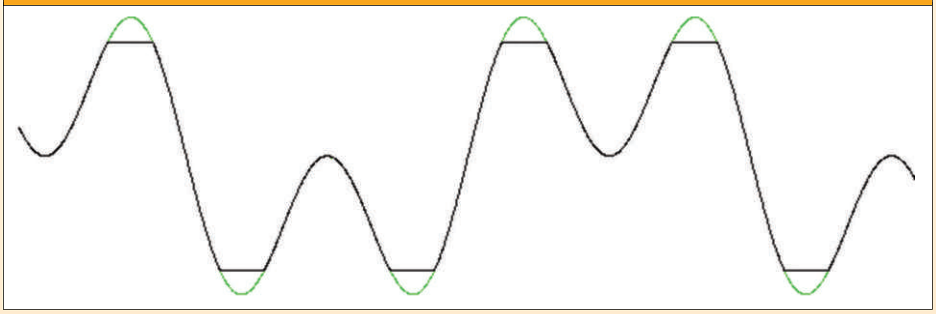
Figure 9. PWM interrupt code.

won't it only match once, so the counter will always be too high? Technically, yes, but because the hardware of the counter will be physically limited to a certain number of bits, the high order bits just disappear and the counter will go back to 0. This is just like the odometer in your car. If you manage to drive it more than 100,000 miles, suddenly you have a brand new car again — according to the odometer, which has gone back to all 0s.

To write some simple code to do PWM, you might make an interrupt routine that triggers when a timer overflows, like what is shown in Figure 9. The reason that you want it in an interrupt is because this frees up your processor to do other things, instead of constantly polling the count value.

If you choose to use a resistor ladder, then you can simply write the value that you want to the I/O port and forget about it. A resistor ladder consumes no processor power. The down side is that it does consume a lot of I/O pins.

Figure 11. Waveform that is clipped.



The Fun Stuff

Now that all of the nitty gritty details are out of the way, let's talk about how to make pleasing sounds come out of the speaker. The first thing

that you may want to try is to generate a sine wave. Although it is quite possible that you could calculate a sine wave as your program is running, that would be a horrible waste of processor power. When working with microcontrollers, it is always a good idea to try to figure out how different types of code will compile. Trigonometric functions take huge amounts of processing power.

A better way would be to make a lookup table of a sine wave. For this example, a 256 value table will be used. The sine wave in the lookup table will be positive and negative, like a real sine wave. In the lookup table, it will vary from –127 to 127.

In our program, we have an interrupt routine that has two 16-bit variables: Counter and Adder. The program will run this routine at regular intervals. Each time that the routine is run, it will add Adder to Counter. The high byte of Counter will be used to find a value from the lookup table. That value will then have 127 added to it to make sure that it is a positive number and then it will be sent to port B, which has a resistor ladder connected to it. If Adder has a low value, then the frequency heard on the speaker will be low. If its value is high, then the frequency heard will be high. If you wanted to vary the volume, you could scale the lookup table result downward to make it quieter or upward to make it louder.

Playing with volume and frequency is nice, but you can step things up another notch by playing multiple frequencies simultaneously. All you will need to do is to have an Adder and a Counter variable for each tone that you want to add to the mix. Take the resulting lookup table values and add them together and then scale the result to meet the needs of the PWM routine or resistor ladder. Be careful to avoid having your result overflow the register that it is stored in. That will make your audio sound terrible. It would be better to have the waveform peak at the top and bottom rather than to overflow.

Figures 10 and 11 show what these waveforms would look like. The green line shows what the intended waveform was. The black line shows the actual outputted waveform. As you can see, clipping still distorts the waveform, but it will sound a lot better than one with overflow errors.

The beginning of this article talked about using a prerecorded sound to indicate that a robot had bumped into something. Recent advances in flash memory have made doing something like this affordable for even a hobbyist. If you have some programming skill on a PC, then making a program to send the information contained in WAV files out of the serial port is a relatively easy task. A WAV file contains

TECH TIDBIT

There is some disagreement about the exact frequencies of the standard, 12 note musical scale, but here is one set of frequencies and the Adder values that you would use to achieve them. If you want to go up an octave, double the Adder value. To go down an octave, divide Adder by two.

Key	Frequency (Hz)	Adder
A	220.0	657
A#	233.1	696
B	247.0	738
C	261.6	781
C#	277.2	827
D	293.7	877
D#	311.2	929
E	329.7	985
F	349.3	1043
F#	370.1	1105
G	392.1	1171
G#	415.5	1241

RESOURCES

www.ccsinfo.com
Sells the C compiler for the PIC processors used in this column.

www.microchip.com
Manufacturer of the PIC microcontroller.

www.digikey.com
Possibly the best source for electronic parts.

a small amount of header information and then the raw sound samples in an uncompressed format. If you use a sound program — such as Sound Recorder, which comes with every copy of Windows — to convert your sound into a mono, eight-bit format, then your program could simply discard the header and send the rest of the file out the serial port to be stored on the flash chip. Your microcontroller would treat the information on the flash chip as one huge lookup table. At regular intervals, it would read a byte and send it to the PWM routine or resistor ladder.

For those of you looking to try out some sound generation on your own, check out Figure 12 for how you might write a program to generate sounds. This program can be compiled by the CCS C compiler and is intended to run on a Microchip PIC16F870 microcontroller with a 20 MHz crystal. It is intended to run with an eight-bit variable voltage resistor ladder connected to port B of the PIC. During testing, it was connected to the input of a home stereo for amplification. A 0.1 µF capacitor was connected between the output of the resistor ladder and ground to smooth out some sound artifacts.

It is interesting to note that the program listed here does not just sweep upward to its highest frequency and then start over at the lowest frequency. What it does instead is to sweep up to a frequency of 11,025 Hz and then sweep back down to its lowest frequency. The reason for this is that the program is hitting the Nyquist limit, defined by its sample rate.

Essentially, what this means is that, for a given sample rate, the maximum frequency that you will be able to reproduce will be one half of the sample rate. As the program tries to output a frequency higher than the Nyquist limit, aliasing occurs and the sound heard is actually lower than intended.

Sound generation is a pretty easy process to accomplish with a microcontroller. With the information presented in this column, you could give your robot a bit of personality by having it share its internal state through beeps or melodies. You could make your house's doorbell sound like a foghorn by having a microcontroller play a WAV file or you could make some sort of musical instrument. How far you want to take it is up to you. SV

Figure 12. Sine wave generation program (also available at **www.servomagazine.com**).

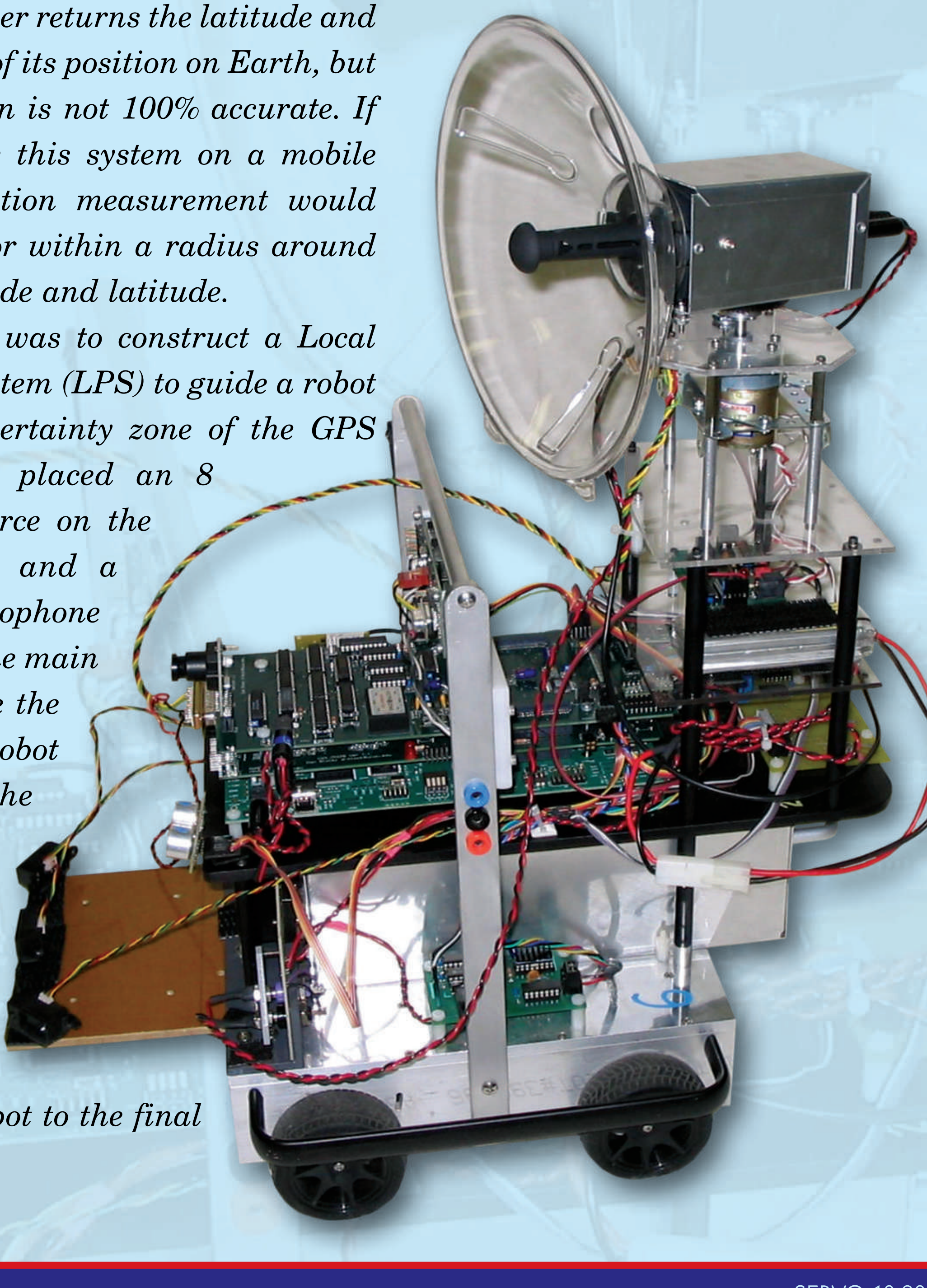
```
// sound02.c
// This program outputs a sine wave of increasing frequency through the output
// of a variable voltage resistor ladder connected to portB.

#include <16F870.h>
#device adc=10
#fuses  HS,NOWDT,PUT,NOPROTECT,BROWNOUT,NOLVP,WRT,NODEBUG
#use fast_io(a)
#use Delay(Clock = 20000000)  // Clock is 20 Mhz
#byte portB = 6

int16 Counter, Adder;

// below is a lookup table that contains 256 values for a sine wave
const signed int8 sinWave[] =
      {0,3,6,9,12,15,18,21,25,28,31,34,37,40,43,46,49,52,54,57,60,63,66,68,71,73,76,
      79,81,83,86,88,90,92,95,97,99,101,103,104,106,108,110,111,113,114,115,117,118,
      119,120,121,122,123,124,125,125,126,126,127,127,127,127,127,127,127,127,127,
      127,126,126,125,125,124,123,123,122,121,120,119,117,116,115,113,112,110,109,
      107,105,104,102,100,98,96,94,91,89,87,85,82,80,77,75,72,70,67,64,61,59,56,53,
      50,47,44,41,38,35,32,29,26,23,20,17,14,11,7,4,1,-1,-4,-7,-11,-14,-17,-20,-23,
      -26,-29,-32,-35,-38,-41,-44,-47,-50,-53,-56,-59,-61,-64,-67,-70,-72,-75,-77,
      -80,-82,-85,-87,-89,-91,-94,-96,-98,-100,-102,-104,-105,-107,-109,-110,-112,
      -113,-115,-116,-117,-119,-120,-121,-122,-123,-123,-124,-125,-125,-126,-126,
      -127,-127,-127,-127,-127,-127,-127,-127,-127,-127,-126,-126,-125,-125,-124,
      -123,-122,-121,-120,-119,-118,-117,-115,-114,-113,-111,-110,-108,-106,-104,
      -103,-101,-99,-97,-95,-92,-90,-88,-86,-83,-81,-79,-76,-73,-71,-68,-66,-63,-60,
      -57,-54,-52,-49,-46,-43,-40,-37,-34,-31,-28,-25,-21,-18,-15,-12,-9,-6,-3,0};

void main() {
   set_tris_b(0b00000000);             // portB is all outputs

   setup_timer_2(T2_DIV_BY_1,75,3);    // sets up the timer to overflow
   setup_ccp1(CCP_PWM);                // ~22050 times a second
   enable_interrupts(INT_TIMER2);      // and turns on the interrupt
   enable_interrupts(global);

   Adder = 0;
   Counter = 0;

   while(true) // do this forever
      {
      // this part makes the sine wave change frequency
      Adder++;
      delay_ms(1);
      }
}

#int_TIMER2
TIMER2_isr() {
// this interrupt routine triggers approximately 22050 times per second
// and is what generates the sine wave.
signed int16 temp;
   Counter += Adder;
   temp = sinWave[*(&Counter + 1)];      // get the lookup table value
   temp += 127;   // add offset to make sure it is a positive number
   portB = temp;
}
```

Sound Source
APPROACHING ROBOT

BY CARLOS MONTESINOS

A GPS receiver returns the latitude and longitude of its position on Earth, but this information is not 100% accurate. If we were to use this system on a mobile robot, the location measurement would include an error within a radius around the real longitude and latitude.

Our project was to construct a Local Positioning System (LPS) to guide a robot within the uncertainty zone of the GPS resolution. We placed an 8 kHz audio source on the desired target and a parabolic microphone on the robot. The main idea was to use the GPS until the robot got near the desired point, then switch to the LPS and use the parabolic microphone to guide the robot to the final position.



Nowadays, GPS is used in airplanes and cars as a navigation aid. Originally developed by the US Army, the Global Positioning System is being enhanced to increase its accuracy. In the past, GPS — without any error correction system — provided a ±15 meter accuracy. The latest update — the WAAS error correction system — provides a ±3 meter accuracy. These values are pretty good for airplanes and even cars, whose average sizes are around 4 and 25 meters, respectively; unfortunately, for smaller mobile robots, this could be a problem.

Let's assume we want to position a robot with a 40 cm width on a 2 m wide bridge. Although we could activate the WAAS system on the GPS, a ±3 error could cause the robot to fall from the bridge. To avoid these errors, we could use a sound device to guide the robot to reach the desired position.

For our experiment, we used a function generator and a speaker to produce the guide sound. A parabolic microphone was mounted on the robot to detect sound signals (Figure 4). To select the signal's frequency, we studied the frequency response of the transducer.

The results showed two peaks: 4 kHz and 8 kHz. We ultimately chose an 8 kHz signal because it gave us fewer headaches.

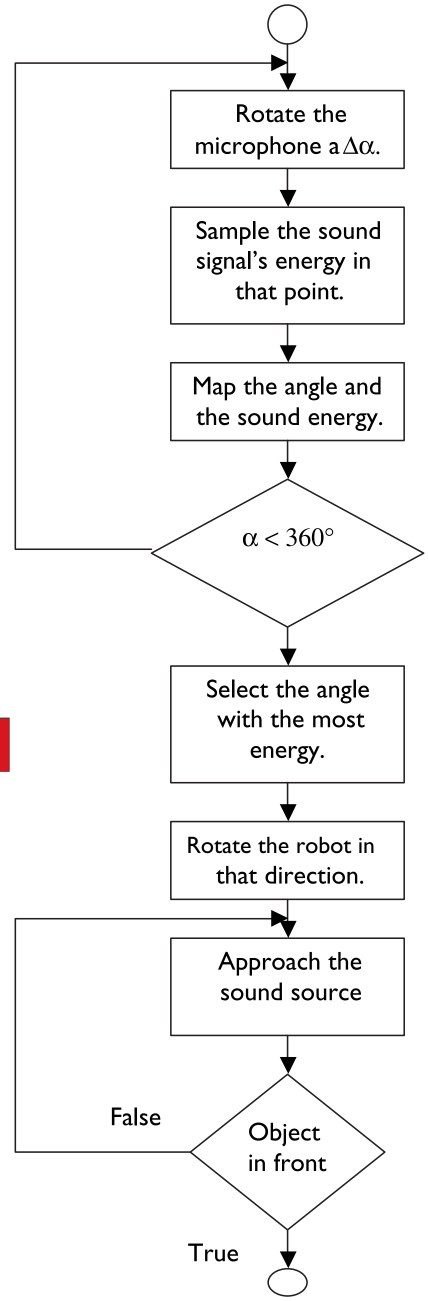


FIGURE 1

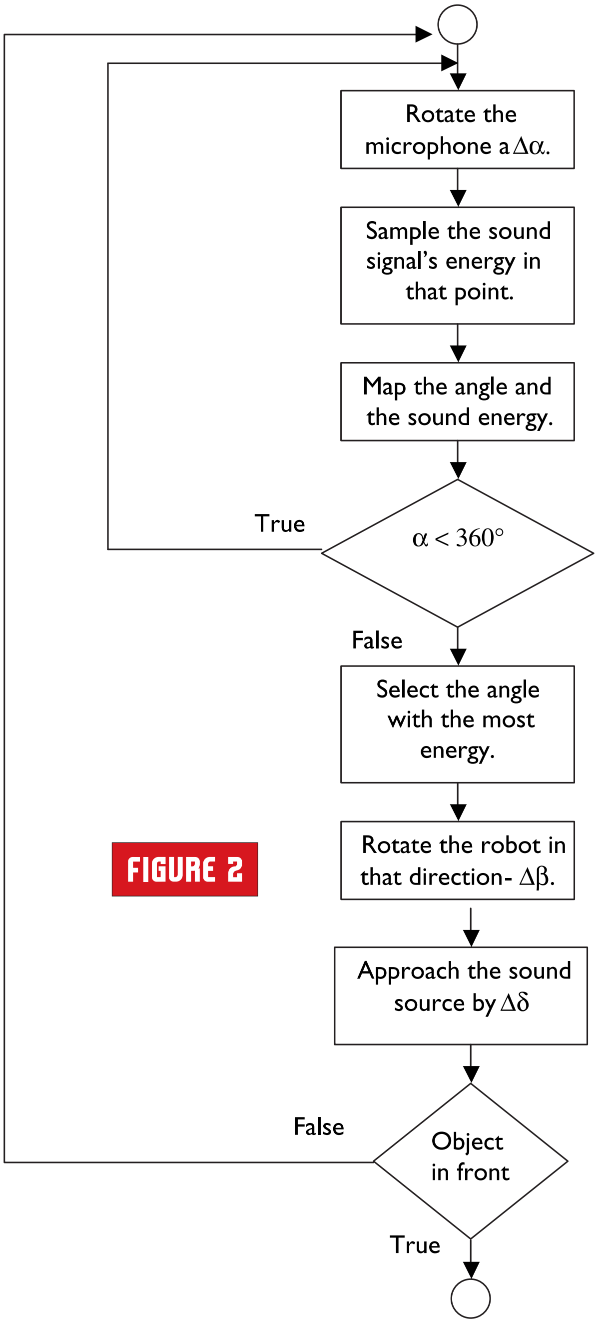


FIGURE 2

Components

For this project, we enhanced one of the existing robots used in our GE330 class. The extra parts for the listening tower and sound source are:

Batteries:
Thunder Power batteries, 2,100 mAh three cell 11.1 V LiPO
Microphone:
ORBITOR, parabolic hearing device
Motor:
HN-GH12-2217Y, 30:1 gear ratio, DC 12 V, torque: 600 g-cm
Optical Encoder:
US Digital S5S-1250-IB8
Switch:
SPDT, contact switch
Function Generator:
HP33120A, arbitrary waveform generator
Speakers:
Sony SRS-A27, active speaker system

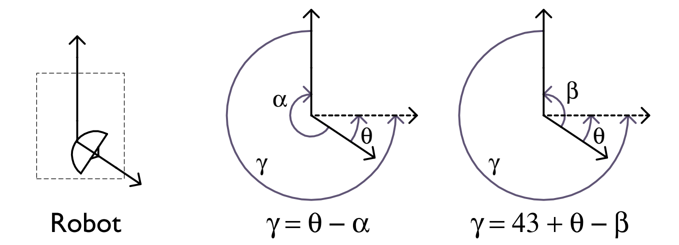


FIGURE 3

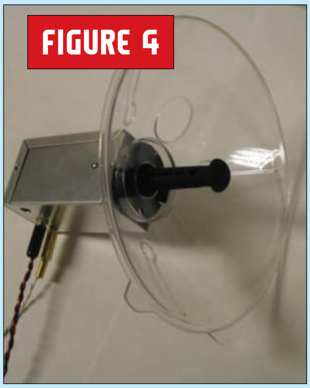


FIGURE 4

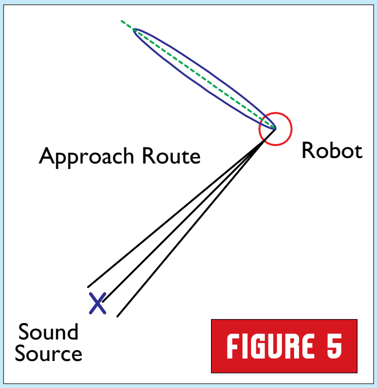


FIGURE 5

Approach

To solve the motion control problem, we used two different algorithms. The first one approached the target directly, while the second one approaches the objective in a tightening spiral pattern. These two algorithms are illustrated in Figures 1 and 2.

Geometric Models:
Figure 3 shows how both the robot's and the antenna's angles are related to each other. Since we are using a gear train between the encoder and the motor to increase its resolution, we found that 2π radians of antenna rotation equates to 43 counts on the encoder. When calibrating the antenna, we had to obtain the phase angle between the robot and the dish.

Since we used an SPDT switch to help us with the calibration process, we had to sample it before we started calibrating. If the switch was set to the left side of the robot, the antenna would rotate clockwise to obtain the phase shift with respect to the red axis.

The opposite is true if the switch was set to the right side of the robot. The linear transformations, seen in Figure 3, take an angle related to the blue axis (the antenna's zero position when the robot is turned on) and map it to the red axis (the robot's zero position).

Geometric Model of Robot Path
Linear Approach:
For this type of approach, the robot will obtain only one sample and use it to find the sound source. Figure 5 shows the path. This path has a key weakness; since the error increases in direct proportion to the distance between the robot and the sound source, one sample is not enough to accurately reach the target.

On the other hand, this path is much faster — compared to the spiral approach — and it could also be used as a first approximation to our target.

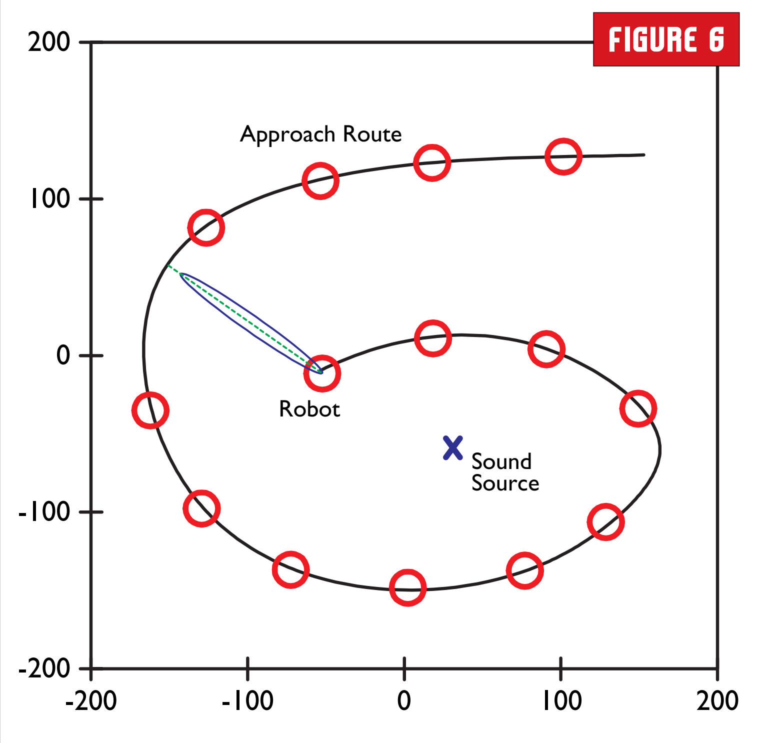
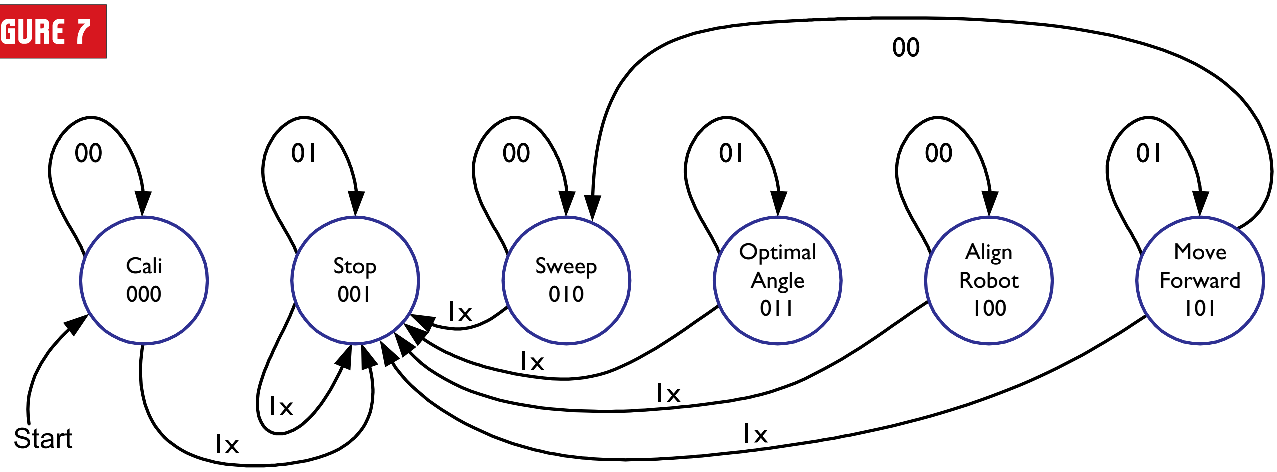


FIGURE 6

FIGURE 7



Spiral Approach:

This path — as opposed to the linear one — will progressively refine the closing vector and decrease position error. In the long run, the robot is expected to reach the target more accurately. This path's great weakness is that it takes the robot longer to approach the sound source, as seen in Figure 6.

Software Structure

Internally, we used a state machine to model the different phases of solving the problem. The state machine is depicted in Figure 7.

The state machine was implemented with a combination of switch statements, as shown in the pseudocode that follows. We used two bits to control the flow, in a variable named inputControlFlag. The most significant bit was used as an emergency escape, while the least significant bit was used to move from one state to the other. The emergency bit can be used to stop the robot if it bumps into or gets too close to an object. In the code, we created macros to replace the use of the numbers shown under each state's title.

This pseudocode shows what is being done in each state of the state machine:

Antenna Calibration:

```
IF The antenna is not calibrated, THEN Calibrate
  GET Where the antenna's switch is initially pointing
  REPEAT
    IF The antenna switch is pointing Right,
      THEN Rotate antenna just a bit in the Left direction
    IF The antenna switch is pointing Left,
      THEN Rotate antenna just a bit in the Right
    GET Where the antenna's switch is pointing now
  UNTIL The antenna's switch differs from the initial
        antenna's switch position
  GET The phase antenna's rotation angle measured from
      the initial position
  SET Calibration comes to an end
ELSE
  CASE We get an antenna's rotation angle sign OF
    Positive:
      SET Equation flag to Positive
      GET Where the antenna's switch is pointing now
    Negative:
      SET Equation flag to Negative
      GET Where the antenna's switch is pointing now
  ENDCASE
  Move ahead in the State Machine
ENDIF
```

Robot Stop:

```
CALL moveRobot() with zero forward turning speeds
WHILE No Move-ahead-to-the-next-State signal
  Wait for a Move-ahead-to-the-next-State signal
ENDWHILE
Move ahead in the State Machine
```

Antenna Sweep:

```
GET Where the antenna's switch is initially pointing
REPEAT
  IF The antenna switch is pointing Right,
```

References

Francesco Bullo, Ph.D.

Dan Block, M.S.

J. Cohen, D. Nichols, and M. Zhou, "Techno Dancing Robot," University of Illinois, Final Project for GE330.

Example C Code for Digitally Processed Audio, by Spectrum Digital Incorporated, 2003.

For Your Info

[1] In this specific project, a full encoder's rotation equivalent value is around 43. See Figure 3 for more details.

```
    THEN Rotate antenna just a bit in the Left direction
  IF The antenna switch is pointing Left,
    THEN Rotate antenna just a bit in the Right
  GET The antenna's angle
  IF phase antenna's rotation angle is Positive
    THEN compute the antenna's angle, related to the
         robot's point of view, with: 2π (or a full encoder's
         rotation equivalent value) 1 + (antenna's angle) -
         (phase antenna's rotation angle)
  IF phase antenna's rotation angle is Negative
    THEN compute the antenna's angle, related to the
         robot's point of view, with: (antenna's angle) -
         (phase antenna's rotation angle)
  GET The power of the sound signal for that position
  IF The power is greater than the past measurement
    THEN Save the antenna's angle as an optimal angle for
         rotating the robot in the future
  GET Where the antenna's switch is pointing now
UNTIL The antenna's switch differs from the initial
      antenna's switch position
 Move ahead in the State Machine
```

Get Optimal Angle:

```
GET Which algorithm to use — Linear or Spiral approach
CASE The algorithm selected has a value OF
  Linear:
    SET The robot's aligning angle equal to
        the optimal angle
  Spiral:
    SET The robot's aligning angle equal to
        (the optimal angle) - (Δα)
ENDCASE
Move ahead in the State Machine
```

Align Robot:

```
REPEAT
  CALL rotateRobot() with aligning angle
UNTIL The robot is aligned
Move ahead in the State Machine
```

Move Robot Forward:

```
CASE The algorithm selected has a value OF
  Linear:
    REPEAT
      Move forward a Δdistance
    UNTIL The robot is close enough to an object
    SET Next state to Stop
  Spiral:
    IF Nothing is in front of the robot,
      THEN Move forward a Ddistance
    SET Next state to Sweep again
ENDCASE
```

Results

In general, the Linear Approach algorithm was tested much more than the Spiral Approach algorithm. This is because the lab's area was not big enough to allow the robot to make big loops while approaching the sound source. The walls of the room also produced undesired sound reflections. In those cases, the robot got confused and started looking for ghost sound sources. Other than the reflections, the robot performed exactly as expected.

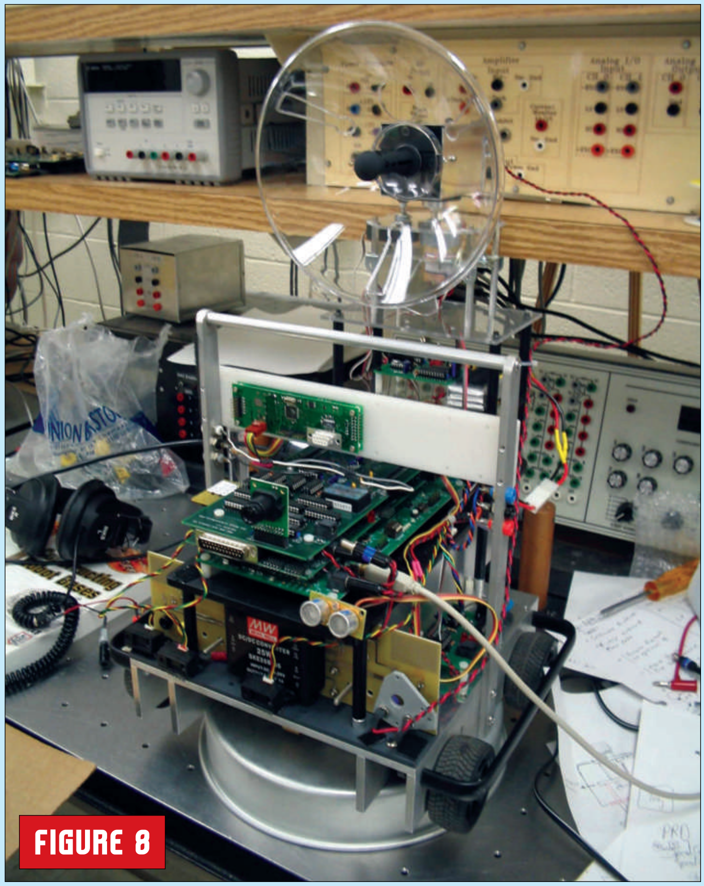


FIGURE 8

Conclusions and Future Enhancements

The system worked pretty well in a small, enclosed lab. Further experiments should be conducted in bigger areas, where reflected signals are minimized. Since the sensor could be activated by sound sources other than the function generator used in the lab, a band pass filter should be integrated. The test frequency used for this experiment was audible. This could become annoying over time, so switching to an ultrasonic frequency might be nice. SV

About the Author

Carlos Montesinos is studying electrical engineering at the Universidad San Francisco de Quito in Quito-Ecuador, South America. He is entering graduate school in a year to continue in the study of robotics.

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

October and November are shaping up nicely this year with seven robot competitions each. There are a few more coming up in December and January, but the Christmas and New Year holiday season tends to be a slow time for robot events.

One event that we were expecting in October has been delayed. The IRRF Open Challenge was conceived as an alternative to DARPA's Grand Challenge for autonomous, all-terrain robots. It was originally planned for this October, but it won't be happening on schedule. The event has been delayed until at least April of 2005, "pending negotiations with the City of Las Vegas, Clark County, and the Bureau of Land Management."

Also, don't forget that October 31st is the deadline for submitting your entry in the *SERVO Magazine* Hack-a-Sapien contest. The Robosapien was a toy designed to be hacked by robot builders, so get to it!

Finally, I'd like to offer my congratulations to the University of Texas at Dallas team for placing ninth out of 18 in their first attempt at the International AUV competition. Good work, guys!

— R. Steven Rainwater

For last minute updates and changes, you can always find the most recent version of the complete Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

October

2 **Robot — Liga**
Kaiserslautern, Germany
Several events. If I could read German, I'd tell you more about them ...
www.robotliga.de/

8-10 **Robot Fighting League National**
Herbst Pavilion, Fort Mason Center
San Francisco, CA
Radio controlled vehicles destroy each other — and bank accounts — in San Francisco.
www.botleague.com/

9-10 **RoboMaxx**
Grants Pass, OR
Includes a range of events for autonomous robots, including maze solving, 3 kg sumo, mini sumo, micro sumo, and nano sumo.
www.sorobotics.org/RoboMaxx/

16-17 **House of NERC**
Frieden's Firehall, Friendens, PA
Radio controlled vehicles destroy each other in the Firehall.
http://robotconflict.com/

21-23 **Tetsujin**
RoboNexus, Santa Clara, CA
SERVO Magazine's weight lifting competition for powered, articulated exoskeletons offers an event incorporating the technology of the future. The event is being held in conjunction with RoboNexus. See page 58 for more information on Tetsujin.
www.servomagazine.com/tetsujin2004/

22-24 **Critter Crunch**
MileHicon, Marriott Southeast, Denver, CO
The Denver Area Mad Scientists were pitting autonomous and remote controlled robots against each other long before commercial events like *BattleBots* and *Robot Wars*.
www.milehicon.org/

27-31 **FIRA Robot World Cup**
BEXCO, Busan, Korea
Check out all the usual categories of robot soccer, including humanoid, single, team, khepera, and many others. Visit the website for further details.
www.fira.net/

31 ***SERVO Magazine*'s Hack-A-Sapien Contest**
This is the deadline for submissions, which should include photos and a written description of your Robosapien hack. May the best hack win!
www.servomagazine.com/hack-a-sapien/

November

6 **CIRC Autonomous Robot Sumo Competition**
Peoria, IL

In addition to sumo, this year's event includes some R/C combat events.
www.circ.mtco.com/competitions/2004/menu.htm

10 **AESS National Robotics Contest**
Barcelona, Spain
Based on the photos available at the website, this is a student contest for line followers and sumo robots.
http://aess.upc.es/concursrobot/

12-13 **Texas BEST Regional Competition**
Moody Coliseum, SMU, Dallas, TX
Students and corporate sponsors build robots from standardized kits and compete in a challenge that changes each year.
www.texasbest.org/

13-14 **Eastern Canadian Robot Games**
Ontario Science Centre, Ontario, Canada
Includes BEAM events, including autonomous sumo and a fire fighting competition.
www.robotgames.ca/

19-21 **All Japan MicroMouse Contest**
Tokyo, Japan
The latest in a long-running series of micromouse contests includes some of the fastest micromouse robots around.
www.bekkoame.ne.jp/~ntf/mouse/taikai/taikai.html

19-20 **South's BEST Competition**
Beard-Eaves Memorial Coliseum, Auburn University, Auburn, AL
Students and corporate sponsors build robots from standardized kits and compete in a challenge that changes each year.
www.southsbest.org/

22 **Texas BEST Competition**
Reed Arena, Texas A & M University
College Station, TX
This is the big one, where the winners from the regionals compete.
www.texasbest.org/

26-27 **War-Bots Xtreme**
Saskatoon Saskatchewan, Canada
Robots (R/C vehicles) attempt to destroy each other to win a whopping $10,000.00 in prize money.
www.warbotsxtreme.com/

December

6 **Hawaii Underwater Robot Challenge**
University of Hawaii, Oahu, HI
This event is a regional for the MATE ROV competition.
www.phys.hawaii.edu/~aapt/calendar/events2003-04.html

11 **Boonshoft Museum LEGO Mindstorms Robotics Competition**
Boonshoft Museum, Dayton, OH
This year's Robotics competition will be a FIRST LEGO League event. The normal FIRST rules apply.
www.boonshoftmuseum.org/special_events.php3

11 **LEGO MY EGG-O Robotic Egg Hunt**
Great Lakes Science Center, Cleveland, OH
A robot egg hunt for students of the Case Western Reserve University Autonomous LEGO Robotics class.
www.eecs.cwru.edu/courses/lego375/egghunt.html

January 2005

24 **Citrus Robotics Robot Combat**
Inverness, FL
Radio controlled vehicles destroy each other in Florida.
www.citrusrobotics.com/

BATTERIES

BATTERY TYPE	Manufacturer	Part Number	Size/Orientation	Voltage	Capacity (mAhr)	Dimensions (mm)	Weight (g)
Alkaline	Energizer	522	9 V	9	595	17.5 x 26.5 x 48.5	45.6
	Energizer	E95	D	1.5	18,000	34.2 dia. x 61.5	141.9
	Energizer	E93	C	1.5	8,350	26.2 dia. x 50.0	6.2
	Energizer	E91	AA	1.5	2,850	14.5 dia. x 50.5	23
	Energizer	E92	AAA	1.5	1,250	10.5 dia. x 44.5	11.5
	Energizer	E90	N	1.5	1,000	12.0 dia. x 30.2	9
	Panasonic	AM-1PI	D	1.5	17,000	33.3 dia. x 61.1	141
	Panasonic	AM-2PI	C	1.5	7,800	25.5 dia. x 50.0	70
	Panasonic	AM-3PI	AA	1.5	2,870	14.5 dia. x 50.5	24
	Panasonic	AM-4PI	AAA	1.5	1,150	10.5 dia. x 44.5	12
	Panasonic	6AM-6PI	9V	9	620	48.5 x 26.5 x 17.5	45
Nickel Cadmium	Panasonic	P-100AASJ/B	AA	1.2	1,080	14.5 dia. x 50.0	23
	Panasonic	P-200SCS	SC	1.2	2,100	23.0 dia. x 43.0	51
	Panasonic	P-500DR	D	1.2	5,500	33.0 dia. x 61.0	145
	Sanyo	N-250AAA	AAA	1.2	270	10.5 dia. x 44.4	11
	Sanyo	KR-1100AAU	AA	1.2	1,100	14.3 dia. x 50.3	25
	Sanyo	RC-2400	SC	1.2	2,300	23.0 dia. x 43.5	60
	Sanyo	KR-5000DEL	D	1.2	5,400	33.2 dia. x 59.5	150
					(10 hr rate)		(kg)
Sealed Lead Acid	Hawker	PC 535	—	12	13 Ahr	170 x 99 x 155	5.44
	Hawker	NP-GEL30-12	—	12	24 Ahr	195 x 133 x 155	10.4
	Panasonic	LC-R067R2P	—	6	6.8 Ahr	151 x 34 x 100	1.26
	Panasonic	LC-R127R2P	—	12	6.8 Ahr	151 x 65 x 100	2.45

ERRATA

The August "Brain Matrix" contained errors regarding two Picobytes products — the PicoPic and the Servio. The correct specifications for these products are as follows; the first specification is for the PicoPic and the second is for the Servio:

Channels: 20, 20; **Controllers per serial bus:** 255, 255; **Servos per serial bus:** 5100, 5100; **Serial connection:** two wire (Rx, Gnd), three wire (Tx, Rx, Gnd); **Serial type:** RS232, TTL level RS232; RS232, TTL level RS232; **Baud rate:** 1200-115200, 1200-115200; **Automatic Baud rate reduction:** no, no; **Pulse width range:** 500-2,400 µs, 750-2,200 µs; **Pulse resolution:** 1 µs, 1 µs; **Position update frequency:** 57, 40; **Velocity control:** yes, yes; **Velocity resolution:** 255, 255; **Acceleration control:** no, no; **A/D inputs:** 0, 8; **Digital I/O:** 20/20, 28/28; **EEPROM/RAM size:** 256, 256; **Number of commands:** 24, 41; **Servo position feedback:** no, yes; **Firmware upgradeable:** yes, yes; **Programmable:** no, yes; **Size:** 1.5 x 2.5, 2.5 x 2.5; **Power requirement:** 5 V @14 mA, 5 V @ 14 mA; **List price:** $49.95, $89.95

MATRIX

by Pete Miles

Upcoming topics include drive motors, radio data links, and SBCs. If you're a manufacturer of one of these items, please send your product information to: **BrainMatrix@servomagazine.com** Disclaimer: Pete Miles, as well as the publishers, strive to present the most accurate data possible in this comparison chart. Neither is responsible for errors or omissions. In the spirit of this information reference, we encourage readers to check with manufacturers for the latest product specs before proceeding with a design. In addition, readers should not interpret the order printed as any form of preference; products may be listed randomly or alphabetically by either company or product name.

BATTERY TYPE	Manufacturer	Part Number	Size/Orientation	Voltage	Capacity (mAhr)	Dimensions (mm)	Weight (g)
Nickel Metal Hydride	Energizer	NH22	9 V	7.2	150	15.9 x 25.2 x 47.5	41
	Sanyo	HR-4U-800	AAA	1.2	800	10.5 dia. x 44.0	13
	Sanyo	HR-3U-2300	AA	1.2	2,300	14.3 dia. x 50.4	30
	Sanyo	RC-3300HV	SC	1.2	3,300	23.0 dia. x 43.5	60
	Sanyo	HR-DU	D	1.2	9,000	34.0 dia. x 59.3	178
	Panasonic	HHR70AAAJ	AAA	1.2	720	10.5 dia. x 44.5	13
	Panasonic	HHR210AA/B	AA	1.2	2,000	14.5 dia. x 50.0	29
	Panasonic	HHR300SCP	SC	1.2	2,800	23.0 dia. x 43.0	55
	Panasonic	HHR650D	D	1.2	6,500	33.0 dia. x 60.8	170
Lithium Ion	Panasonic	CGR18650C	Cylindrical	3.6	2,150	18.6 dia. x 65.2	44.5
	Panasonic	CGA103450	Prismatic	3.6	1,950	34.0 x 50.0 x 10.5	39
	Sanyo	UR14500P	Cylindrical	3.7	720	13.8 dia. x 49.1	19.5
	Sanyo	UR18650F	Cylindrical	3.7	2,100	18.1 dia. x 64.7	46.5
	Sanyo	UF611948P	Prismatic	3.7	420	5.6 x 19 x 47.5	11.5
	Sanyo	UF103450P	Prismatic	3.7	1,700	10.5 x 33.8 x 48.8	39.5
Li Polymer	Sanyo	UPF385269		3.7	1,150	3.8 x 25 x 68.5	27
			Power				(kg)
Fuel Cells	FCS	555-530116	1 W	0.6	—	25 x 60 x 60	0.13
	FCS	555-530107	10 W	6.5	—	80 x 70 x 70	1.13
	FCS	555-541510	100 W	11.5	—	150 x 120 x 120	2.72
	FCS	555-540310	1 KW	16	—	140 x 240 x 240	13.6

by James Antonakos

THE ASSEMBLY LINE

The Plan Is Hatched

The robot assembly line is empty. There are no parts in the store room. There are no instructions to follow to build anything. All because there is no plan. Before we begin throwing parts together to build a new robot, we must have a plan. The plan will enable us to construct the robot in order to achieve the design goals. The design goals are chosen to satisfy a set of requirements. If we are building the robot for sale, the requirements will be specified by the customer. If we are building the robot for ourselves, we choose the requirements depending on what we want to do with the robot. This is where we will begin our journey.

In 1950 and 1951, a neurophysiologist named W. Grey Walter wrote two articles for *Scientific American* magazine in which he described the design and operation of two simple robots (which Dr. Walter called tortoises). These robots rolled around on the floor, seeking a light source; in this pursuit, they used the ability to adjust their path when encountering an obstacle. What could be simpler than that? This will be the initial goal of our new design, replicating a 1950s era design with updated hardware (and software, as well). So, we have two basic requirements for our new robot design:

1. The robot must have the ability to move and change direction on a smooth, level surface, as it seeks a light source.

2. The robot must have the ability to change its motion when encountering an obstacle.

These two requirements bring a number of questions to mind. First, what kind of light source will be used? Some light sources — such as an ordinary light bulb — emit their bright light over a wide area. Other light sources — such as LEDs — have a much smaller viewing angle and typically much less brightness. This may force the robot to be very close to the LED or in front of it to be able to sense it.

How do we sense the light source? The electronic sensor used will depend on the choice of light source. A photocell

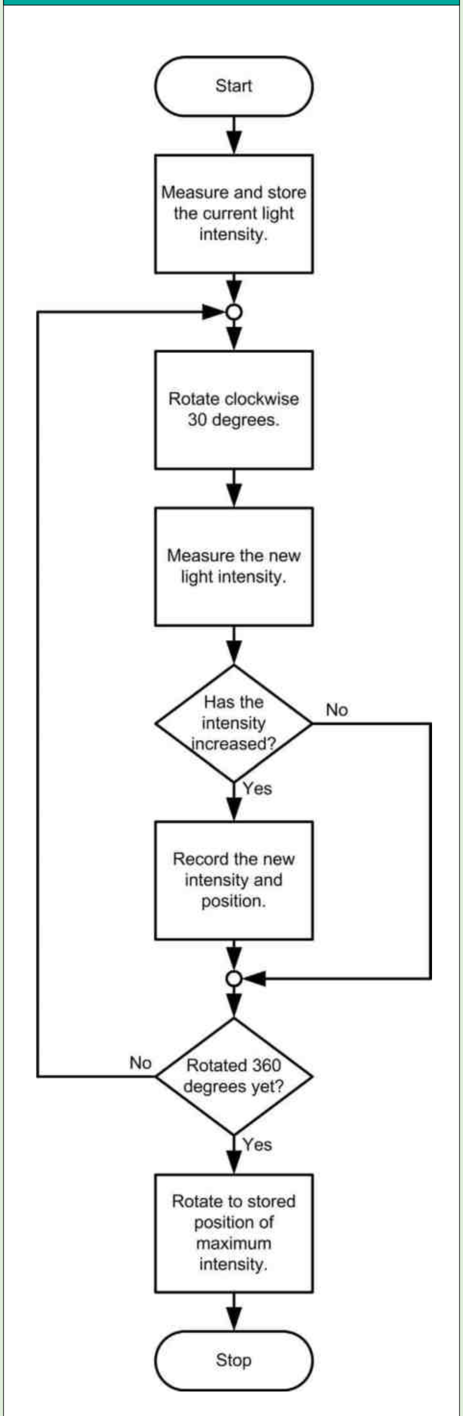
would be a good choice for an incandescent light source. A phototransistor would be better when an LED is used. We may even choose to use an infrared light source, which would then require an infrared detector. Infrared is invisible to the ordinary eye, which may lead to difficulty during troubleshooting. How do we know if the infrared light source is working? We cannot see whether it is off or on. We would have to look for a signal at the infrared sensor output to determine its status. So, even our simple choice of a light source impacts the design, troubleshooting, and operation of the robot.

Next, how do we get the robot to move? Will it crawl around on six legs or use wheels or treads? If legs are used, do they all operate independently or will there be connected pairs that cooperate? How many legs will be needed? When using wheels, will one or more be mounted on a swivel to allow for turning as the robot moves? Will forward and reverse be needed or just forward? What type of motors should be used (stepper, servo, or ordinary DC motor)? How fast should the robot move? As before, we make our choice depending on different factors. One important factor will be the size and weight of the robot. A large, heavy robot may require a bigger servo motor tather than a stepper motor (due to the torque differences between them). The size of the robot depends on its application and the physical dimensions of the components within the robot.

Finally, how does the robot know it has encountered an obstacle? We may use a bumper fitted with a microswitch, a simple mercury-filled tilt switch, a pair of ultrasonic transducers to measure distance (anything within a certain distance is a "collision," even if the objects have not touched yet), or we may sense the motor current and look for an increase when resistance from an object is encountered and the motor has to work harder.

The behavior of our robot must be designed to assist in meeting the goal of locating the light source and moving closer to it. This could be a two-step process or a continuous process. In the first step, the robot could scan the environment, looking for the maximum light intensity as it rotates in-place in one direction. The second step involves moving forward, toward the light source once the proper direction is located. In the continuous process, the robot makes constant changes to its direction as it seeks the

Figure 1. Flowchart showing how the maximum light intensity position is located.



maximum light intensity.

The flowchart in Figure 1 shows one way to perform the first step of locating the position of maximum light intensity. The flowchart is not perfect. In fact, there is an error in the logic, but it is an error of omission. The position of maximum intensity is found by rotating clockwise in 30 degree increments, but what if something is blocking the robot from rotating clockwise? Then we have a problem.

Often, these types of problems do not show up until you are testing the design with the actual robot (unless you have developed a software simulator along with the hardware). You may want to practice rewriting the flowchart to utilize rotation in both directions with input data from a collision detector.

All of this activity needs some kind of control circuit. Dr. Walter's robots utilized two vacuum tubes to form a simplistic analog computer to adjust the drive motors according to measured light intensity. While we could update the design by using transistors, a better approach would be to use a microcontroller with the appropriate interface and driver circuitry. The microcontroller will allow us to write software to implement the scanning loop from Figure 1, as well as add new features to the robot with a minimum of fuss.

Our plan must not only specify how to build the new robot, but also suggest possible future improvements and capabilities. For example, a future goal may be to develop a second robot and add communication capability to the robots. Each robot would map the environment and transfer its data to the other robot. Additional behaviors may include:

- Only returning to the light source base when battery power is low
- Returning to a specific light source base when multiple bases are used
- Searching a maze
- Following a pre-planned route
- Responding to spoken commands

Before finishing, one last item needs to be addressed. What are we going to call the new robot? After all, it will be an intelligent being, interacting with its environment, and it will have a personality unique to itself.

For lack of something better, let us call the robot Uno (Number One). Like proud parents, we will be able to watch Uno grow, acquire new abilities, and expand its horizons. Next time, we will examine the actual mechanical and electrical design of Uno and see how many of the questions posed here will be answered. SV

TERMS

LED (Light Emitting Diode)
A special diode that gives off visible light when properly biased.

Microcontroller
Essentially, a computer on a chip, containing the processor core, memory, digital and analog I/O, timers, and other functions.

Photocell
A light-sensitive resistor. As light intensity increases, it's resistance decreases.

Phototransistor
A light-sensitive transistor. As the light intensity increases, so does the collector current.

Servo motor
A motor whose position can be controlled very accurately through the use of a feedback mechanism and a control circuit. Normally, the entire system is simply referred to as a "servo."

Stepper motor
A motor whose position can be controlled fairly accurately by stepping the shaft in small rotational increments.

Ultrasonic transducer
Devices similar to speakers and microphones, except they emit and sense high frequency signals above the range of human hearing.

ABOUT THE AUTHOR

James Antonakos is a Professor in the Departments of Electrical Engineering Technology and Computer Studies at Broome Community College. You may reach him at antonakos_j@sunybroome.edu

FIRST FRENZY 2004 PART 2

BY JUSTIN LYONS

My name is Justin Lyons. I am a junior at Chaparral High School in Temecula, CA and have been an active member of FIRST Team 1079 CREATE for the past two years. I hope to use the skills I have been honing through Club CREATE's activities to help me reach my aspiration of attending Cal Maritime Academy. I would also like to earn a degree in engineering. My hobbies — in addition to robotics — include sailing, road biking, paint ball, and the violin.

During the 2004 FIRST competition, I volunteered to lead the project focused on the mechanism to manipulate the large multiplier ball. It was a fascinating and difficult task and proved to be one of the more complex aspects of the build. I enjoyed the opportunity to challenge myself and work on a critical and defining part of the robot.

As you may already know, For Inspiration and Recognition of Science and Technology (FIRST) is an organization that seeks to inspire young people to get involved in the areas of robotics and technology. This high school robot-

Standing, from left to right: Bryce Woolley, Jairus Ciocon, Bryant Nelson, Kristen Baber, Justin Lyons, Ryan Potts, Jack Gordon. Kneeling: Evan Woolley.

Team members not pictured: Evan Heid, Josh Larsen, Austin O'Neal, Ivan Hoffman.

JUSTIN & JACK

ics competition features a different task each year for teams throughout the world to accomplish in the allotted six weeks. The 2004 challenge suggested that the teams devise a lifting and grasping mechanism that could maneuver balls, which, in turn, affect the score of both your opponent, as well as your own team.

The first step in developing the appropriate mechanism that would be capable of lifting and gripping a 30-inch diameter rubber ball was to get the whole team, as well as all of our mentors, together to brain storm. While sitting in the Woolley family's living room with some of our mentors from the University of California, Riverside, we all had a chance to share our ideas with each other. Although many people's thoughts and ideas were shot down or discredited almost before they could be shared (which ended up causing much of the tension that occurred during the build), we were able to agree on three design possibilities.

The first one involved the use of a pneumatic piston to actuate two arms that would tightly grasp the ball. This idea seemed almost flawless, until we took into consideration the size of the compressed air reservoirs with which we had to work. We would also be using pneumatics to actuate the ball kicker that would be mounted on the front of our 130 pound robot. In addition, we had to consider the amount of weight we would have hanging off of the front of the robot, affecting its center of gravity (CG). Too much weight that far out in front of the robot — combined with the weight of the ball — could cause the robot to tip forward and over, thereby disabling us for the remainder of that round.

The second idea was to use two small motors that would actuate two large claws and grab the ball; this was similar to the idea with the two pneumatic pistons. This idea fell through because we had a limited number of vectors to control the motors. If we were to buy the two additional vectors that would have been needed, it would have cost us over $300.00 — money we didn't have at the time.

The third idea — the one that we ended up going with — seemed almost too simple. It was a basic scoop that would fit between the PVC pipes of the goal and lift the ball out of it. All it would require was a small aluminum frame and possibly two small pneumatic cylinders that would actuate two finger-like structures to help hold the ball in the scoop. These cylinders would be actuated only once or twice throughout the whole match.

Once we had narrowed our choices down to these three, we took a vote among team members for which idea would be used in the competition to be held at the Los Angeles Sports Arena in late March. The vote concluded that we would be using our third idea (The Scoop) for our final design. Some team members took it harder than others when their ideas were not used on the robot, but — after a while — they acquiesced.

Finally, we had a preliminary design for the mechanism that would hold the ball. Jack Gordon — a fellow team member — and I made a crude mock-up of what we wanted the scoop to look like. This allowed us to get a better idea of what the dimensions of the scoop should be, according to the size of the ball. Using simple 1" x 2" wooden stock, we accomplished this task.

This mock-up was instrumental in the final development of the arm that would make or break our success in the upcoming 2004 competition. We used and abused it many times throughout the build, testing different locking mechanism ideas that would hold up the arm. Strangely, it was not necessary to have a locking mechanism at the beginning of each match. We later discovered that the arm could be positioned resting against the robot itself, meeting the initial size requirements before each match.

This mock-up also aided in our design for the ideal lengths of each of the "fingers" of the arm that would help support the ball once it was resting on the arms.

Once Jack and I were sure of the dimensions of the arm, we decided that we would use 1" x 1" aluminum box channel with a 1/16" wall thickness. We went to purchase the required materials, but learned that our local hardware store didn't have the channel we needed with a 1/16" wall thickness, so we went with a 1/8" wall thickness. This increase in thickness and weight to the arm was not enough to cause any problems.

We purchased 20' of this stock for about $8.00, which was reasonable and within our already tight budget. Only about 16' was necessary at the end of this project, but the additional 4' was added for potential adjustments

JACK & BRYCE

that could arise as we drew closer to the final product.

When cutting the aluminum, we made sure to follow Mr. Woolley's "Golden Rule" — always measure twice and cut once! This advice proved true when some of the team members failed to follow this important rule, thereby causing delays in valuable time and wasting materials. Yet, we still somehow managed to accomplish our task.

Once we had cut all of the parts for the arm, we set out to find a local welder who was gracious enough to donate the time, skill, and materials to our project. Finding this local welder proved to be no easy task. We must have called seven or more local businesses before we were able to find one that was open. It was Saturday at the time our search began and time was of the essence; we could not wait until Monday. Mike's Precision Welding of Temecula, CA became our "Knight in Welded Armor" by coming to our rescue.

As anyone who has ever owned a small business knows, time is money and even one day can affect your business. The time donated by this company and others was greatly appreciated and valuable to the completion of this project. Thanks again to Mike's Precision Welding!


TEAMWORK


MORE FAB

Now that the arm was created, it was necessary to connect it to the robot itself. It wasn't clear whose idea it was, but — at some point — it was decided that we would need to fasten a mast-like rod to the front of the robot to support the arm we had created. We did this using a 4' x 1" x 2" aluminum box channel rod fastened to the front frame of the robot using 1/4 -20 machine bolts and nylocks.

Once we had figured out how the mast would be attached to the robot, we had the task of attaching the arm to the mast. In order for the arm to prove beneficial in a match, it would have to be able to get under a 30" diameter rubber ball resting atop eight four-foot vertical PVC pipes, which were positioned and fastened in a circle onto a flat base with wheels. These PVC pipes held the rubber ball like fingers.

With the understanding of this challenge ahead of us in the competition, it was necessary to attach the arm to the mast in a manner that would allow the robot to collect the 30" ball from the finger-like PVC pipes. Jack was assigned this task. Once Jack had finished it (one week later), we called it "Jack's Contraption." He had devised a sliding mechanism that would be capable of easily accomplishing this task. Although slightly overbuilt, it worked perfectly (after some shaving, filing, and swearing) when it came down to the competition.

The sliding mechanism was built using a 6" x 3" x 2" piece of aluminum "U" channel, along with two aluminum "L" brackets at 6" x 1" x 1" to add support to the piece of "U" channel, which was only about a 1/16" thick. As if that wasn't already enough, he also added a 1/4" thick 7" x 4" aluminum plate on the back. To allow the sliding mechanism to move up and down the aluminum mast easily, Jack used some HDPE (High Density Polyethylene) that we had left over from the 2003 FIRST Competition. This was all held together by small machine bolts and Nylocks. Once assembled and attached to the arm and the cables that ran to two window motors, "Jack's Contraption" ran up and down the mast of our robot smoothly.

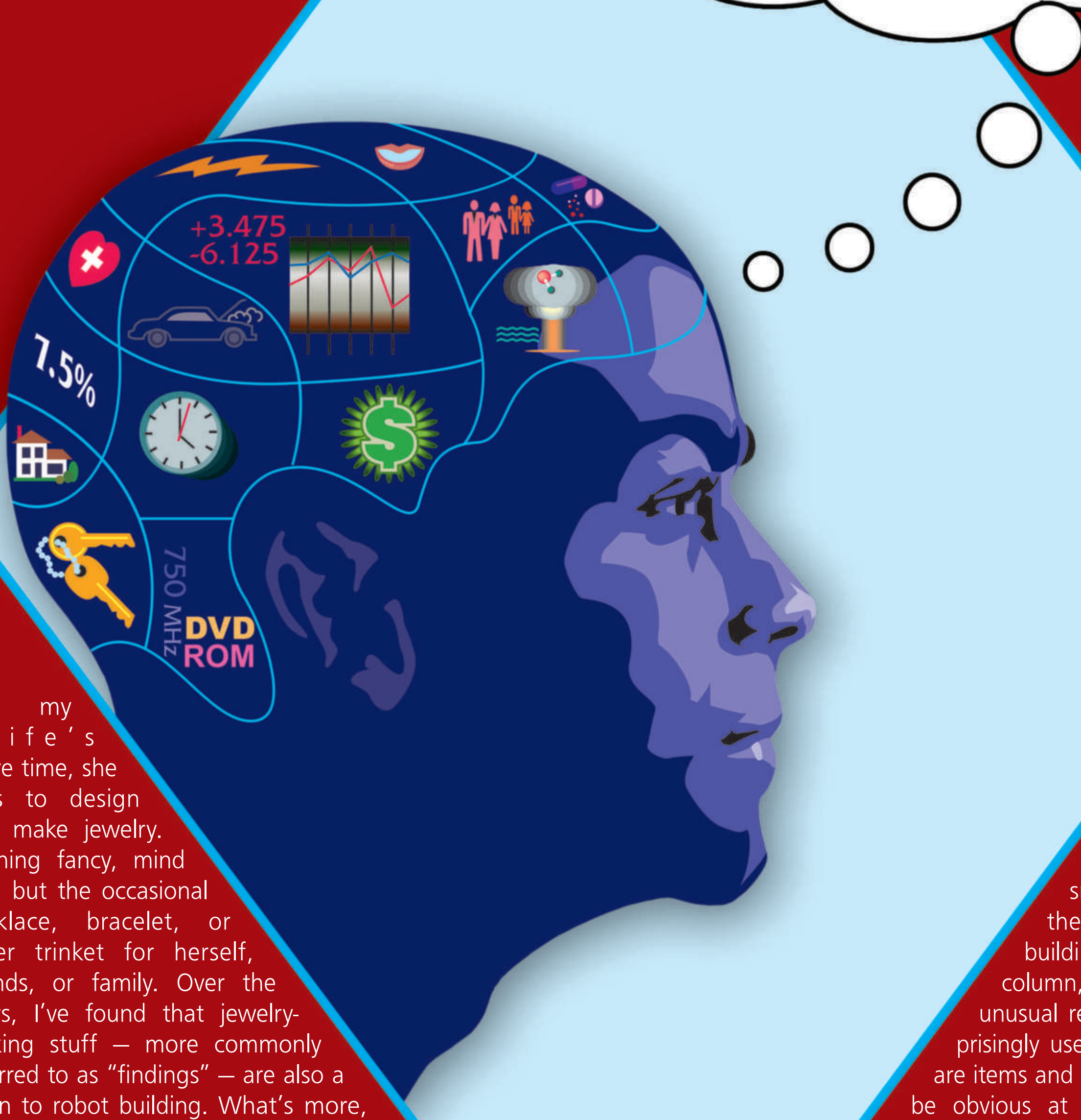
A few days before we had to ship our robot off to a storage facility to await the Regional Competition in Los Angeles, our team attended a sectional practice competition at Chatsworth High School, put on by FIRST Team #22. This was a good chance for most of the Southern California teams to work out the kinks in both their mechanical, as well as electrical, systems on their robots.

While at the sectional, we made many important discoveries concerning the strength of the pneumatic fingers that were mounted on the end of the arm. Due to a mix-up in the controls of the robot, the fingers slammed directly into the lexan and diamond plate wall, completely destroying one of them and bending the other. This was a good thing for us to find out before the actual competition. After we overcame that obstacle, we didn't have any more major problems. Our team won every match we were placed in. All of us were extremely satisfied with the performance of the robot.

As one might realize by now, the design and construction process takes much time, thought, and effort. It is a collaborative effort that does not completely end once the structure has been built. There is always something to be found that could be fine-tuned or honed to perfection, which we discovered at the competition. Stay tuned for more on the trials and tribulations of the 2004 FIRST Robotics Competition. SV

by Gordon McComb

Thinking Outside the Box



In my wife's spare time, she likes to design and make jewelry. Nothing fancy, mind you, but the occasional necklace, bracelet, or other trinket for herself, friends, or family. Over the years, I've found that jewelry-making stuff — more commonly referred to as "findings" — are also a boon to robot building. What's more, many online and local stores that cater to the hobby jewelry-making crowd also sell useful crafts items that are likewise handy as robot construction pieces. As long as it's not solid gold or diamond studded, most jewelry and crafts findings are both cheap and versatile. Things like pin backs for broaches can be used as brackets or extremely fine, but super-strong, wire that can be used for pulley cable. Also, there are always blinking light strands and the guts to electronic musical toys in the crafts section.

It's often the small things that make the big differences in building a bot. So, for this column, I will concentrate on unusual resources for some surprisingly useful robot parts. These are items and sources that might not be obvious at first glance, but will undoubtedly become regular choices once you get to know about them. You'll need to "think outside the box" to visualize how some of the products listed here can be used in robotics, but the effort is well worth it.

Some obvious sources that we won't bother to talk about this time around include automotive supply stores, discount and department stores, dollar stores, and R/C hobby stores. Several of these have been

covered in detail in past "Robotics Resources" columns, either in *SERVO Magazine* or in *Nuts & Volts*. Check your back issues. Because of the number of resources to cover, we'll split this discussion into two parts. See next month's "Robotics Resources" for the conclusion of "Thinking Outside the Box!"

Transfer Films

Laser and inkjet printer transfer films make color decals, printed circuit board layouts, signs, control panels, and more.

Bel, Inc.
www.beldecal.com

Bel markets an extensive line of custom decals, including ink jet decals for doing it yourself. The ink jet (also works with many laser printers) decal film is the water slip type; after printing, place it in water for a short period of time, then transfer it to a flat, clean surface — wood (sanded, closed grains are best), plastic, or metal. They even sell a "tattoo paper" that lets you make temporary tattoos of your rad robots.

Dyna Art/Pulsar
www.pulsar.gs

Makes and sells a PCB transfer system; uses laminator-like machine to fuse artwork (prepared by laser printer) onto copper clad.

HPS Papilio
www.papilio.com

Manufacturers of water-slide decals for laser and ink jet printers, as well as specialty laser and ink jet papers and coatings, including adhesive-backed ink-jet vinyl, printable magnetic media, temporary tattoo papers ("Tobor Loves Mom"?), and self-adhesive window film.

Image Solutions
www.bestimagesolutions.com

Distributes heat transfer (or digital transfer) supplies, vinyl media, water slide decal media, and bumper sticker media. Can be used to customize robots, create labels, and control panels, etc.

Lazertran Limited
www.lazertran.com

Lazertran is a thin paper that is used to transfer full-color images to flat surfaces, such as wood, tile, metal, or plastic. You first print the image onto the paper using a color photo copier. The paper is then left to soak in water — like a decal — until its emulsion is soft. The emulsion is then transferred to the surface you want to cover. You can use Lazertran in lieu of painting or as labeling for a control panel.

McGonigal Paper & Graphics
www.mcgpaper.com

Online retailers of specialty papers for arts and crafts, including: water slide decals, glow-in-the-dark transfer film, super Color Shrink (shrinks when heated), black light film (fluoresces under ultraviolet light), and window cling decals (clings to glass and other very smooth surfaces).

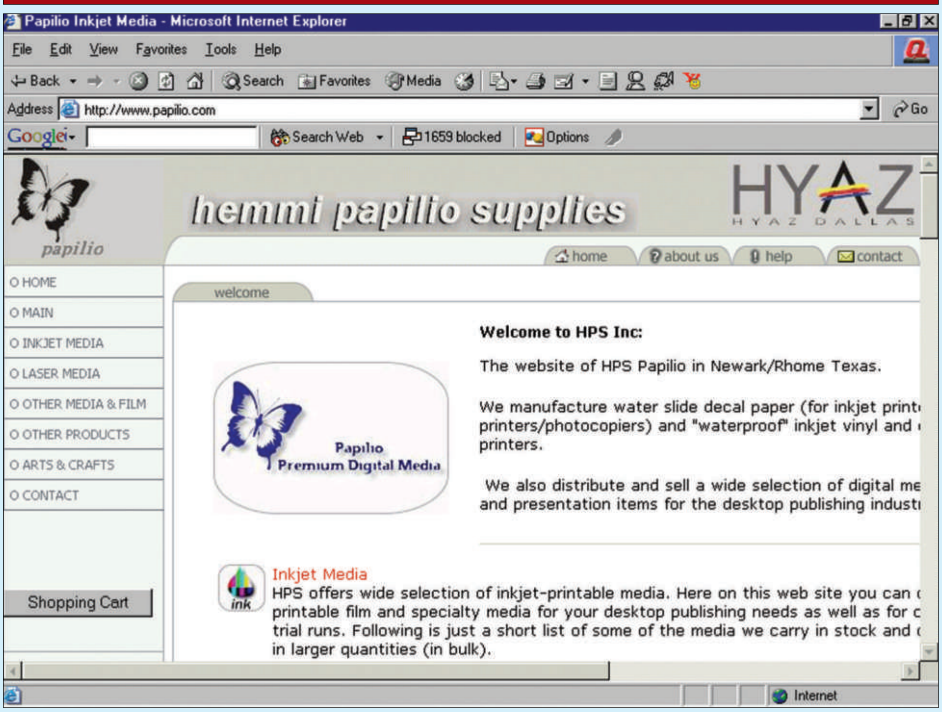
SuperCal Decals
www.supercaldecals.com

SuperCal makes and markets a line of water transfer decal sheets that can be printed on using an ink jet printer. Print your design, soak it in water, and transfer the design to plastic, metal, wood, and many other non-porous surfaces. Online sales are available, but the transfer sheets are available at many hobby stores.

Visual Communications
www.visual-color.com

Makers of Mirror Image iron-on transfer papers. Available for copiers, laser printers, and ink jet printers. For color or black and white. Suitable substrates include many woods, metal, plastic, and fabric. Also sells presses, heat press stand, and transfer tools.

FIGURE 1. Papilio inkjet papers and films.



Store Fixtures

I know it's strange, but retail store fixtures can be used to build robots. Store fixtures include:

- One- to two-inch diameter lightweight PVC and aluminum tubing, used to create displays and racks. Along with the tubing are various types and styles of connectors: two-, three-, four-, five-, even six-way connectors — all of which can be used to build cheap (as in inexpensive) robot frames.

- Ball bearing Lazy Susan turntables, intended to make rotating displays, but useful for any rotational movement in a robot. Sizes from 3 to 12 inches in diameter and even larger.

- Slat wall is used to mount peg bars for holding merchandise. This material is plastic with T-shaped slots grooved into it

at even intervals. With simple hardware, you can use Slat wall as a reconfigurable robot base, no drilling required.

Alpha Store Fixtures, Inc.
www.storefixtures2000.com

Store fixtures: slat wall, body forms, acrylic plastic bins and trays.

Display Warehouse
www.displaywarehouse.com

Store fixture components: pope, clamps, slat board, etc.

Tebo Store Fixtures
www.tebostorefixtures.com

A place to buy store fixtures. What you're looking for here are things like tubing, slat wall, and other building materials for your robots.

WR Display & Packaging
www.wrdisplay.ca

Store supplies and display fixtures. Some interesting products include:

- Jewelry boxes — great holders for small parts.
- Slatwall and accessories — possible use in making robot bases.
- Display hardware — for making robot frames.

Paper and Plastic Laminates

A laminate is a sheet composed of two materials glued together, almost like a sandwich.

Laminates save money by using only a small amount of the expensive outside material, relying on the inside material to represent most of the bulk. Because two kinds of materials are cemented together, they tend to reinforce one another.

A commonly available laminate is foam core, available at most craft and art supply stores. The material is composed of compressed Styrofoam (or similar material), lined on both sides with heavy, colored paper. You can cut foam core with a knife or a small hobby saw and ordinary paper glue can be used to hold the cut pieces together. Form core is the most common paper laminate, but it's not the only one by far. There is a wide variety of paper and even plastic laminate sheets available and many brands are listed here. Some manufacturers of paper and plastic sell directly, but most want you to purchase their goods from arts and crafts retailers. For those makers that don't sell directly, you can see their websites for handy application notes and material handling datasheets.

Advantage Distribution
www.advantagedistribution.com

Plastic products, including acrylic sheet, expanded foamed PVC (Sintra), corrugated plastic. Also sells Bienfang Foamboard, Pillocore Foamcore, and Ultra Board (polystyrene foam core).

GoldenWest Manufacturing
www.goldenwestmfg.com

Manufacturer of casting resins (both rigid and flexible), machinable plastic, foam board, and tools for cast and mold making. Their machinable plastic includes a product known as Butter-Board, a lightweight plastic block that is non-abrasive and very easy to machine or work with hand tools.

Public Missiles, Ltd.
www.publicmissiles.com

Public Missiles sells parts and electronics for model rocketry. They offer some nice polymer tubing (strong but light), wrapped phenolic tubing, electronic altimeters, two-part expanding foam, and other odds-and-ends that an Earth-based robot ought to find a use for.

R & J Sign Supply
www.rjsign.com

R & J Sign specializes in materials for sign-makers. Of particular interest to us robot constructors is Sign Foam; they also sell several other light-weight — yet strong — substrates that can be used to build machine bodies and other parts. Sign Foam is a rigid, high-density urethane material available in several different weights. It's easy to cut and can be "sculpted" to various shapes. Also sells Alumalite (aluminum over foam), corrugated plastic, and PVC foam board.

ULINE
www.uline.com

Think "out of the box" on this one — literally out of the shipping box. ULINE caters to people who ship things by mail or freight. That means cardboard boxes ... and lots of other interesting things, like foam padding, tubes, shrink wrap, and tons of other stuff. ULINE is one of the largest mail room supply companies and they'll send you a full-color catalog in a heartbeat. Some interesting products you might want to consider, for robot construction, for shipping, or for the workshop:

- Antistatic packing materials (foam, tubing, shielding bags, and bubble

pack)
- Foam for shipping boxes
- Bubble bags and sheets
- Clear and colored mailing tubes, with and without end-caps
- Little clear plastic boxes (great for building small modules)

Jewelry Findings

Jewelry findings are small plastic or metal pieces used to make jewelry. Several common types are useful in the construction of smaller robots. For example, earring posts and earring backs can be used to make miniature linkages. Small beads can be used as spacers and even bearings; jump rings can be used to attach small parts. (They're bendable with pliers.) Spacer bars can be used as tiny linkages.

Darice, Inc.
www.darice.com

Manufacturer of jewelry findings and arts/crafts products. Catalog (450+ pages) costs $20.00. Darice sells their product through arts and crafts stores or online.

Fire Mountain Gems
www.firemountaingems.com

Small precision tools and jewelry supplies. Get their printed catalog, which should fire up your imagination. Wholesale prices, but you don't have to buy in volume.

Jewelry Supply
www.jewelrysupply.com

Online source for jewelry making, crafts supplies, and miniature tools. Check the section on jewelry findings. Great source for tiny parts and tools.

Notions Marketing
www.notions-marketing.com

Distributor of a broad line of notions to sewing and craft stores. Publishes a huge printed catalog. There is a $250.00 minimum opening order, so your best bet is to find their wares at local fabric and craft stores.

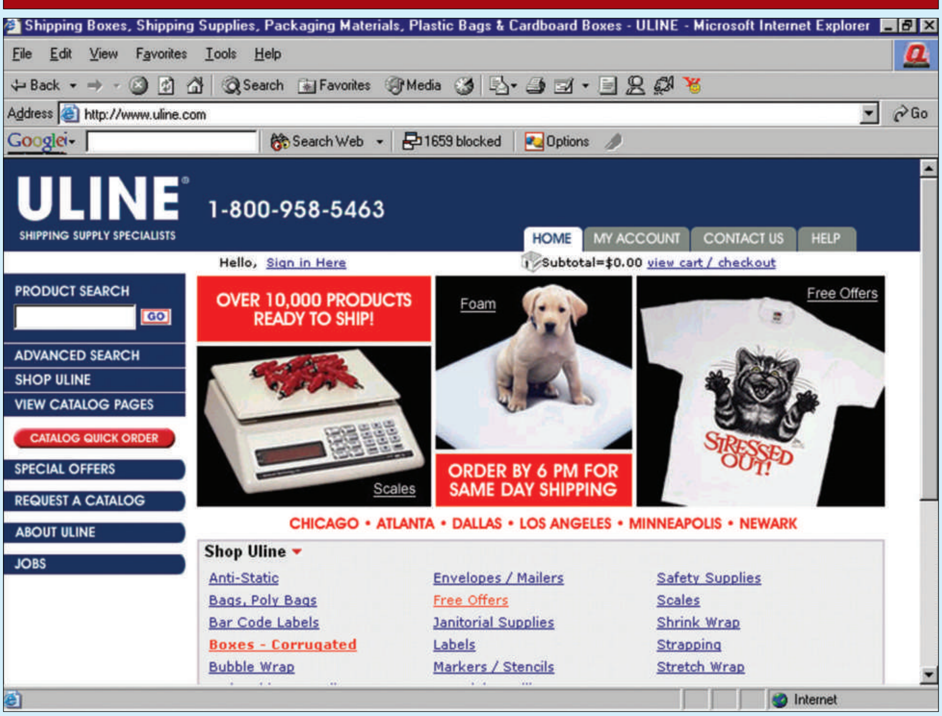
Westrim Crafts
www.westrimcrafts.com

Distributor of craft-making products, including things like jewelry findings. These can be used for construction of small parts. Westrim sells exclusively to retailers; most any local or online craft store will carry their product.

Wrights
www.wrights.com

Manufacturer of sewing and craft notions, including fusing tape (heat it up and it congeals with a stickiness). The products are available at fabric and craft stores.

FIGURE 2. ULINE shipping supplies.



Wood Craft

Wood crafter's supply outlets provide all sorts of interesting tidbits to the robot builder. First, of course, is the wood. Exotic woods are often far stronger (and more attractive) than the regular softwoods — even oak — sold at the local home improvement stores. Yet, in small pieces, they're still rather affordable. Many wood craft outlets also offer small wood pieces you can use for your projects, like jointing pins and "biscuits," quality miniature tools, modern adhesives, and much more.

Craft Supplies USA
www.woodturnerscatalog.com

Of particular interest are the company's wood blanks. Most of these are fairly exotic woods, but they're not all that expensive. A benefit of these exotics is that many are extremely hard and/or dense (like carob) and are a lot stronger than a piece of pine or some other common wood. You might want to consider one of these exotics if you are building a robot with wood and need strength for the base, risers, or some other component.

Smaller variations of these blanks can be found in the pen blanks category and include wood, celluloid, and something called Environ — a manufactured material that looks like granite or wood, but is actually newsprint and soybean by-products. Pen blanks are usually about 3/4 inch by 5 inches.

HUT Products
www.hutproducts.com

Wood, tools, and supplies for the precision woodturning crafter. They offer:

- Wood and synthetic pen blanks — small blocks of wood (usually about 3/4 by 5 inches) intended for making pens using a lathe. You can use the blanks for anything. Pen blank material is typically strong and dense and is ideal when you want structural strength in some part of your robot.

- Woodturning and metal lathes — resellers of the Sherline lathes, as well as the VEGA Mini Lathe Duplicator.
- Acrylic rod — clear and colored.
- Dyed plywood — small sheets (11 by 12 inches) of Baltic birch plywood, 0.350 inch thick.

Stockade Wood & Craft Supply
www.stockade-supply.com

Wood parts — lots of 'em. Includes wheels, pins, biscuits (oval-shaped thin wood for parts and shims), cubes, dowels, and lots more. Most of the products are used to make small, wooden toys or as miniatures for doll houses, but have obvious uses as robo-bits.

Woodcraft Supply Corp.
www.woodcraft.com

Woodworkers' tools and supplies. Be sure to check out their extensive line of plywoods (if you're building a robot base using wood). Of course, they offer the regular hand tools — like drills, saws, and planes — for working with wood. Order online or visit one of their 61 retail locations.

Science

Here, you will find retail outlets for amateur scientific kits, demonstrators, and components. Some of the resources listed here supply science and lab equipment to schools and there may be restrictions on selling directly to the general public. Check the ordering terms for any purchasing limitations.

Analytical Scientific, Ltd.
www.analyticalsci.com

Scientific goodies that would make even Mr. Wizard flush with joy, Analytical Scientific carries chemistry, astronomy, anatomy, and biology kits and supplies. They offer the full line of OWI robot kits and other knick-knacks of interest to automaton builders.

Arbor Scientific
www.arborsci.com

Science and educational materials for teachers and hobbyists. Product categories include: force and motion, light and color, electricity, chemistry, astronomy, measurement, lasers, holography, sound and waves, magnetism, fiber optics, books and videos, science toys, science software, and computer data loggers. Among Arbor Scientific's more notable products that are useful in robotics are: color filters kit (large sheets of colored gel filters in primary and secondary colors), color filter swatch book (small pieces of dozens of colored gels), spring scales (for measuring motor force), laser pointers (hackable laser diodes); helical spring "Snaky" (long acoustic spring, for possible use in bumper detection), sound pipes (medium diameter flexible tubes, for use in construction or looks), and magnets (rare earth, ceramic, and others, for use in sensors, construction, etc.).

Carolina Science & Math
www.carolina.com

Carolina sells a massive amount of educational supplies and materials for schools and researchers. Their (online and print) catalog contains a number of kits and products in the fields of physics, technology ed (including electronics and robotics), and lab equipment/supplies. The company's robotic offerings are the OWI robot kits.

Educational Innovations, Inc.
www.teachersource.com

Science supplies, kits, and demonstrators. Among their useful products for robotics are: CLIMBaTRON window climbing robots, polarizing filters, refracting, diffracting and reflecting lights, magnets, ferrofluids, and nitinol memory metal/Muscle Wire.

Efston Science
www.e-sci.com

Science kits and supplies. Includes mechanical and physical science, astronomy, kits for kids, science fair projects, Jensen tools.

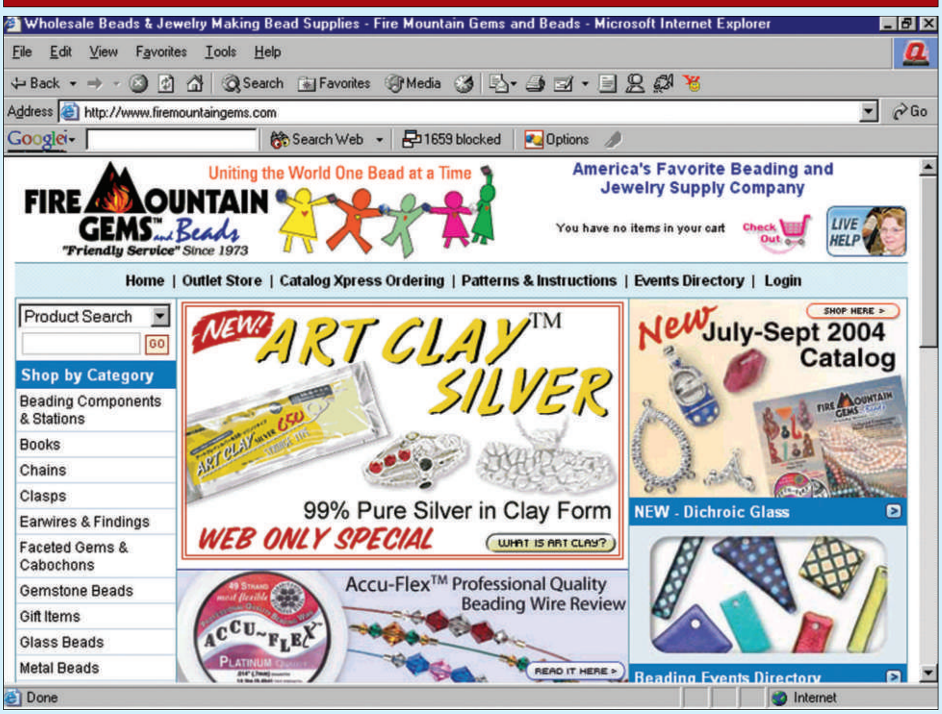
Indigo Instruments
www.indigo.com

Science kits and supplies (test tubes, etc.). Specializes in organic chemistry parts and kits. Some products — like the 3 mm diameter rare earth Neodymium magnets — are useful in robotics sensors.

Kelvin
www.kelvin.com

Kelvin sells educational kits and materials for the high tech teaching world. They also offer project materials in metal, plastic, and wood, along with magnets, various sizes and types of gearboxes (and motors with and without gearboxes), motor holders, linear actuator motors, wheels, gears, sprockets

FIGURE 3. Fire Mountain Gems jewelry supplies.



and sprocket chains, and hundreds of additional products.

Pitsco
www.pitsco.com

Online shopping through their e-commerce portal, **www.shop-pitsco.com** Among the more esoteric offerings useful in robotics are:

- Space Wings electronics kit (shape memory alloy)
- Servo power transmission (hub mounts, sprockets, sprocket chain, gears, wheels)
- Plastic injection molding tools and supplies
- Plastic vacuum forming tools and supplies
- Aircraft birch plywood

School-Tech, Inc.
www.school-tech.com

Science kits — the physical science line of kits includes magnetism, electricity, and robotics (the latter in the form of the OWI robot kits).

Science Kit & Boreal Laboratories
www.sciencekit.com

Selling some 15,000 items, Science Kit specializes in products for education and many of their offerings are packaged for demonstration and multi-student exploration. This can be a good thing; many of their products are "samplers" with a little bit of a lot of things. This can save you from buying larger quantities of individual parts when you only need a little bit yourself.

Scientifics Online
www.scientificsonline.com

Of this company's product lines, the following are of keen value to robot builders:

- Laser pointers — hack these for various projects or make a Borg Bot
- Metal detectors — treasure-hunting robot or maybe a robot that is guided by metal
- Optics — every robot needs glasses for eyes — plus diffraction gratings, filters
- Unique lighting — includes live wire (electroluminescent wire)
- Magnets — rare earth and not-so-rare earth (Alnico and ceramic)
- General science — motors, gears, robots, Fischertechnik, K'NEX Ultra
- Tools — mostly hand tools

The Science Source
www.thesciencesource.com

Designed for the upper-grade science teacher and, if that's what you are, you probably already know about this place, but everyone should know about it, as they have truly unique products (many are intended for classroom demonstrations or group study) that have definite applications in amateur robotics.

Restaurant Supplies

You may be wondering why I listed several online and local restaurant supply outlets. No, I'm not suggesting you build a mechanical Emeril or that amateur robots make for good chefs; personally, I get enough robotic service

at the local fast food joint. Rather, restaurant supply outlets tend to be excellent sources for materials at low cost. Typical restaurant supply outlets carry such items as:

- Stainless steel, spun bowls (miniature dipping and finger-bowl size, up to gargantuan 100 pound mixing bowls).
- Aluminum and steel baking sheets, with and without anti-stick coatings (can be a cheap alternative to aluminum or metal sheet purchased at hardware and hobby stores).
- Kitchen and baking utensils, in plastic, metal, or wood. Cut them to various shapes to make parts for your robots.
- Miscellaneous unusual items, including strainers, individual bread and cake pans (good as the "mother mold" for making small castings), silverware containers that make for great vertical parts bins, and plastic tumblers can be sawed to various sizes and shapes and used for protective robotic covers, shells, and other applications.

When shopping the restaurant supply store, you must think outside the box. Don't look at a miniature plastic salt shaker as a dispenser of sodium chloride. Instead, look at it as a housing for a robotic sensor or the leg tips to a medium-sized walking robot.

Ace Mart Restaurant Supply
www.acemart.com
Online store and chain of local restaurant supply stores in Texas. Look for inexpensively priced metal bowls, mixing utensils, and interesting kitchen gadgets that can be hacked to make things for your robot.

BigTray, Inc.
www.bigtray.com
Online restaurant supply retailer. Cheap source of stainless steel items. Check the "Smallwares" section.

Insta-Wares
www.instawares.com
Kitchen and dining supplies (among other products) for the institutional food service business.

Mission Restaurant Supply
www.missionrs.com
The gamut of restaurant supplies, from small one ounce stainless steel sauce cups to large dough mixers.

***Final note**: Steer away from the retailers that specialize in "gourmet kitchens." Their products tend to be more expensive. The typical restaurant supply outlet sells cheap, because eateries are like any other business — they're always looking for ways to trim costs. You get to enjoy those cost savings, too.* **SV**

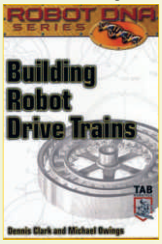
ABOUT THE AUTHOR

Gordon McComb is the author of the best-selling *Robot Builder's Bonanza* and he operates a small manufacturing company dedicated to low cost amateur robotics, **www.budgetrobotics.com**

Mind Candy For Today's Roboticist

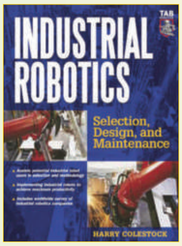
The *SERVO* Bookstore

Building Robot Drive Trains
by Dennis Clark / Michael Owings



This essential title in McGraw-Hill's *Robot DNA Series* is just what robotics hobbyists need to build an effective drive train using inexpensive, off-the-shelf parts. Leaving heavy-duty "tech speak" behind, the authors focus on the actual concepts and applications necessary to build — and understand — these critical, force-conveying systems. **$24.95**

Industrial Robotics
by Harry Colestock



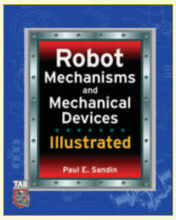
With so many industries taking advantage of the tremendous advances in robotics, entities ranging from small family businesses to large corporations need assistance in the selection, design, set-up, maintenance, and economic considerations of industrial automation. *Industrial Robots* shows how to achieve maximum productivity with robotics, classifies robots according to their complexity and function, and explains how to avoid common automation mistakes. **$39.95**

Robots for Kids
Exploring New Technologies for Learning, First Edition

Edited by Allison Druin / James Hendler

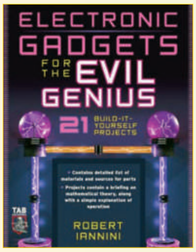
Robots for Kids: Exploring New Technologies for Learning opens with contributions from leading designers and researchers — each one offering a unique perspective into the challenge of developing robots specifically for children. The second part is devoted to the stories of educators who work with children and use these devices, exploring new applications and mapping their impact. Throughout the book, children's essays are provided, discussing their first-hand experiences and ideas about robots. This is an engaging, entertaining, and insightful book for a broad audience — including HCI, AI, and robotics researchers in business and academia, new media and consumer product developers, robotics hobbyists, toy designers, teachers, and education researchers. **$50.95**

Robot Mechanisms and Mechanical Devices Illustrated
by Paul Sandin



Both hobbyists and professionals will treasure this unique and distinctive sourcebook — the most thorough — and thoroughly explained — compendium of robot mechanisms and devices ever assembled. Written and illustrated specifically for people fascinated with mobile robots, *Robot Mechanisms and Mechanical Devices Illustrated* offers a one-stop source of everything needed for the mechanical design of state-of-the-art mobile 'bots. **$39.95**

Electronic Gadgets for the Evil Genius
by Robert Iannini



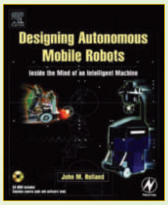
The do-it-yourself hobbyist market — particularly in the area of electronics — is hotter than ever. This book gives the "evil genius" loads of projects to delve into, from an ultrasonic microphone to a body heat detector, all the way to a *Star Wars* Light Saber. This book makes creating these devices fun, inexpensive, and easy. **$24.95**

Robot Builder's Sourcebook
by Gordon McComb

Fascinated by the world of robotics, but don't know how to tap into the incredible amount of information available on the subject? Clueless as to locating specific information on robotics? Want the names, addresses, phone numbers, and websites of companies that can supply the exact part, plan, kit, building material, programming language, operating system, computer system, or publication you've been searching for? Turn to the *Robot Builder's Sourcebook* — a unique clearinghouse of information that will open 2,500+ new doors and spark almost as many new ideas. **$24.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Designing Autonomous Mobile Robots
by John Holland



Designing Autonomous Mobile Robots introduces the reader to the fundamental concepts of this complex field. The author addresses all the pertinent topics of the electronic hardware and software of mobile robot design, with particular emphasis on the more difficult problems of control, navigation, and sensor interfacing. Its state-of-the-art treatment of core concepts in mobile robotics helps and challenges readers to explore new avenues in this exciting field. The accompanying CD-ROM provides software routines for the examples cited, as well as an electronic version of the text. **$49.95**

PIC Robotics: A Beginner's Guide to Robotics Projects Using the PIC Micro
by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU! In this heavily-illustrated resource, the author provides plans and complete parts lists for 11 easy-to-build robots — each with a PICMicro brain. The expertly written coverage of the PIC Basic Computer makes programming a snap — and lots of fun. **$19.95**

Mobile Robotic Car Design
by Pushkin Kachroo / Patricia Mellodge

This thoughtful guide gives you complete, illustrated plans and instructions for building a 1:10 scale car robot that would cost thousands of dollars if bought off-the-shelf. But, beyond hours of entertainment and satisfaction spent creating and operating an impressive and fun project, *Mobile Robotic Car Design* provides serious insight into the science and art of robotics. Written by robotics experts, this book gives you a solid background in electrical and mechanical theory, and the design savvy to conceptualize, enlarge, and build robotics projects of your own. **$29.95**

To order call 1-800-783-4624 or go to our website at www.servomagazine.com

PIC Microcontroller Project Book

by John Iovine

The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, in the four years that have passed since the book was first published, the electronics hobbyist market has become more sophisticated. Many users of the PIC are now comfortable paying the $250.00 price for the Professional version of the PIC Basic (the regular version sells for $100.00). This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which one serves better in different situations. **$29.95**

Robot Builder's Bonanza

by Gordon McComb

Robot Builder's Bonanza is a major revision of the bestselling bible of amateur robot building — packed with the latest in servo motor technology, microcontrolled robots, remote control, LEGO Mindstorms Kits, and other commercial kits. It gives electronics hobbyists fully illustrated plans for 11 complete robots, as well as all-new coverage of Robotix-based robots, LEGO Technic-based robots, Functionoids with LEGO Mindstorms, and location and motorized systems with servo motors. **$24.95**

Robotics Demystified

by Edwin Wise

There's no easier, faster, or more practical way to learn the really tough subjects. McGraw-Hill's *Demystified* titles are the most efficient, intriguingly written brush-ups you can find. Organized as self-teaching guides, they come complete with key points, background information, questions for each chapter, and even final exams. You'll be able to learn more in less time, evaluate your strengths and weaknesses, and reinforce your knowledge and confidence. **$19.95**

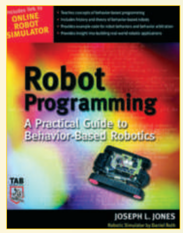
Build Your Own Humanoid Robots

by Karl Williams

Build Your Own Humanoid Robots provides step-by-step directions for six exciting projects — each costing less than $300.00. Together, they form the essential ingredients for making your own humanoid robot. **$24.95**

Robot Programming

by Joe Jones / Daniel Roth



Using an intuitive method, *Robot Programming* deconstructs robot control into simple and distinct behaviors that are easy to program and debug for inexpensive microcontrollers with little memory. Once you've mastered programming your online bot, you can easily adapt your programs for use in physical robots. **$29.95**

Machine Nature: The Coming Age of Bio-Inspired Computing

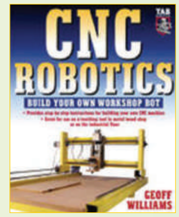
by Moshe Sipper

Despite being marvels of complexity and human ingenuity, computers are notoriously bad at learning new things and dealing with new situations. Researchers at the frontiers of computer science have turned to nature for solutions to the problem of machine adaptation and learning. By applying models of complex biological systems to the realm of computing machines, they have given rise to a new breed of adaptive software and hardware. In *Machine Nature*, computer scientist Moshe Sipper takes readers on a thrilling journey to the terra nova of computing to provide a compelling look at cutting-edge computers, robots, and machines now and in the decades ahead. **$24.95**

Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

CNC Robotics

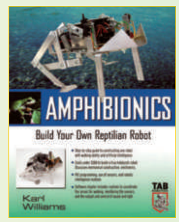
by Geoff Williams



Now, for the first time, you can get complete directions for building a CNC workshop bot for a total cost of around $1,500.00. *CNC Robotics* gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot and can be customized to suit your purposes exactly, because you designed it. **$34.95**

Amphibionics

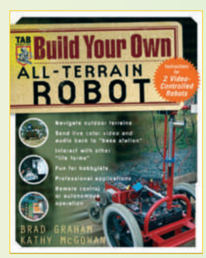
by Karl Williams



If you're a robotics hobbyist with a flair for creativity, here's your opportunity to join the revolution and advance robotic evolution. This work provides the hobbyist with the detailed mechanical, electronic, and PIC microcontroller knowledge needed to build and program snake, frog, turtle, and alligator robots. It focuses on the construction of each robot in detail and then explores the world of slithering, jumping, swimming, and walking robots — and the artificial intelligence needed with these platforms. Packed with insight and a wealth of informative illustrations, *Amphibionics* focuses on construction details and explores the artificial intelligence needed to make these specialized movements happen. **$19.95**

Build Your Own All-Terrain Robot

by Brad Graham / Kathy McGowan

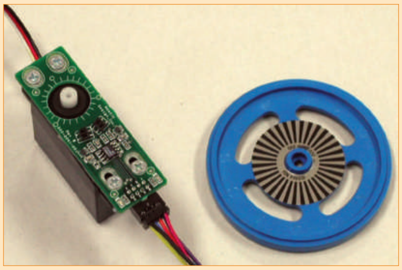


Remotely operated robots are becoming increasingly popular because they allow the operators to explore areas that may not normally be easily accessible. The use of video-controlled technology has sparked a growing public interest not only in hobbyists, but also in the areas of research, space, archeology, deep sea exploration, and even the military. Inside *Build Your Own All-Terrain Robot*, the writers enable even total newcomers to robots to construct a rugged, video-controlled, talking, seeing, interacting explorer bot with a range of over a mile for under $200.00! **$29.95**

New Products

CONTROLLERS & PROCESSORS

WheelWatcher WW-01



Noetic Design, Inc., has introduced its first product — the Nubotics WheelWatcher WW-01 — an incremental encoder kit for use with R/C servo-based robots using standard, injection-molded wheels.

The WW-01 WheelWatcher incremental encoder system attaches to the outside of a standard R/C servo and utilizes an adhesive-backed reflective codewheel designed for standard injection-molded robot wheels to determine wheel rotation rate and distance traveled. It produces standard ChA/ChB raw quadrature outputs, as well as decoded clock and direction signals.

The reflective codewheel offers 32 high contrast stripes. The hardware quadrature decoder used provides a 25 microsecond pulse at each transition of ChA and ChB, resulting in 128 counts per wheel rotation.

Mounting hardware is included. Example programs for many popular robot controllers are available at **www.nubotics.com**

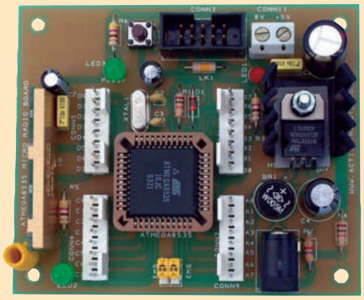
The WW-01 is distributed by Acroname, Inc., which offers single WW-01 kits for $32.95 and a kit of two for $59.95. Contact Acroname for resale and OEM pricing.

For further information, please contact:

Noetic Design or **Acroname, Inc.**	Website: **www.nubotics.com** or Website: **www.acroname.com**

Circle #110 on the Reader Service Card.

WMB-USA — Versatile Microcontroller Board



WMB-USA is a new wireless controller board that uses Atmega8535-16 — one of the latest, fastest Atmel Mega chips, running at 16 MHz and achieving almost 16 MIPS. The built-in serial RF link has variable power from 1 to 10 mW, with selectable data speeds. All of the parameters at the PC end can be easily modified with the configuration software supplied free with the board.

The combination of high speed and seamless wireless communication via your PC or laptop makes the radio micro board particularly well suited for the development of low cost wireless control, mobile robots, and data logging applications. At 16 MIPS, it is many times faster than PIC solutions and there are many compilers available that are suitable for this board.

The easiest to use is BASCOM AVR — a free demo version of this and BASCOM Basic is supplied and compiles to machine code, so there's no loss in speed. Individual applications can easily be developed

in Basic or C.

WMB-USA is only 84 x 70 x 20 mm in size and accepts any input voltage from 6-35 V, with polarity inversion protection too — and 750 mA is available for peripheral electronics needs. Four eight-bit I/O ports and one serial port make the board extremely versatile in terms of control and I/O capability.

WMB-USA Features:

- ATMEGA8535 (Atmel AVR series, 16 MHz (16 MIPS)) 8 Kbyte ISP flash, 512 bytes SRAM, 512 bytes EEPROM, three timers, ADC eight channel, USART, four PWM channels, RTC, WDT with internal RC oscillator, and more
- Pre-assembled board
- ISP port
- ISP download indicating LED
- 32 I/O port pin
- Reset button
- Free Windows Basic/C compilers
- ISP downloader (optional)
- Supplied with PC radio link software

Made in Europe, WMB-USA is available from stock at $199.00 each from Saelig Co., Inc.

A datasheet is available at **www.active-robots.com/products/controllr/micro-radio-details.shtml**

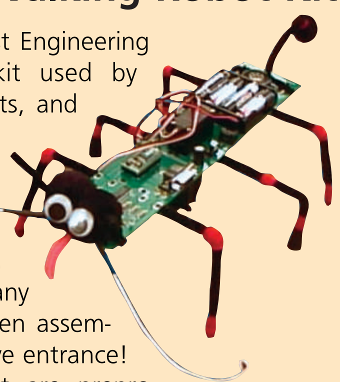
For further information, please contact:

Saelig Company, Inc.
1160-D2 Pittsford-Victor Rd.
Pittsford, NY 14534
Tel: **585 • 385 • 1750** Fax: **585 • 385 • 1768**
Email: **info@saelig.com**
Website: **www.saelig.com**

Circle #130 on the Reader Service Card.

ROBOT KITS

BugBrain™ : A Walking Robot Kit



The BugBrain™ from Yost Engineering is a walking robot kit used by schools, students, hobbyists, and gadget lovers worldwide. The kit is both fun and educational, allowing full programming via a serial connection to any PC. At 15 inches long when assembled, it makes an impressive entrance!

Unlike other kits that are preprogrammed or have a limited instruction set, BugBrains includes a full-featured BasicX microprocessor (BASIC Stamp compatible) and expansion connector to allow users to progress from beginner to advanced skills. Sample programs allow beginners to get started easily, programming the BugBrain to walk, chirp, dance, sing, wink, blink, and sense things in the environment, using the LEDs, speakers, bump sensors, and other kit components.

BugBrains teaches not only programming fundamentals, but also problem solving, critical thinking skills, computer technology fundamentals, and many hands-on technical and fabrication skills. The BugBrain kit comes with all of the necessary mechanical and electronic components, a detailed full-color fabrication/instruction manual, a PC programming cable, all the necessary software, and even an AC adapter to save battery power while perfecting your programs.

Complete CoursePaks are also available, facilitating the kits' use in classrooms, extracurricular enrichment programs, scouting groups, and home schools. CoursePak lab units cover basic electronics and components, programming skills, and other related concepts; also included are quizzes, self-test exercises, and optional exercises.

The kit is available for $169.00 ($122.00 without the microprocessor). Call for school discounts.

For further information, please contact:

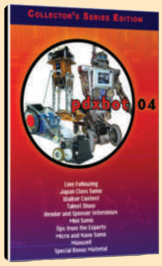
Yost Engineering, Inc.
Tel: **888 • 395 • 9029**
Website: **www.BugBrains.com**

Circle #144 on the Reader Service Card.

Circle #153 on the Reader Service Card.

VIDEOS & DVDs

PDXBOT.04 DVD



Lemon Studios announces the release of *PDXBOT.04* the DVD. *PDXBOT.04* is the Fifth Annual Autonomous Robotics Contest and Expo held June 6, 2004, in Portland, OR on the campus of Portland State University. It was put on by PARTS (Portland Area Robotics Society) and the PSU IEEE Robotics and Automation Society. Lemon Studios broke out all our cameras and filmed the whole event.

Robot enthusiasts from all over the Pacific NW and Canada converge on Portland each year to compete, share ideas, and have a fantastic time checking out the latest robot merchandise. They come to view the cool displays set up by PSU and corporate sponsors — such as Tektronix — and try their hands at interactive demonstrations and perhaps compete for fame and glory. The video includes competition action in beginning line following, advanced line following, walking contest, a robot talent show, Japan class (3 kg) sumo, mini sumo, and even micro and nano sumo.

Catch every exciting moment of competition action, as well as interviews and commentary with well-known pioneers, vendors, and people of interest in the robotics community, such as Bill Harrison, Pete Skeggs, and Monty Goodson. They explain recent new products from Megabitty (BittyBot), Tigerbotics, Nubotics, and others through bonus material and extended interviews. Also on hand are representatives of Parallax, Tigerbotics, Powell's Books, Solarbotics, Medonis Engineering, Southern Oregon Robotics, PSU Engineering, and PARTS.

Enjoy learning about the roots of sumo in Japan and how mini sumo came to the Pacific NW and then went on to become the growing hobby that now has contests in almost every major city in the world. Learn tricks and secrets from the experts and find out about new products and techniques being used here, at the birthplace of low cost competition robotics in the US. This DVD will interest the newcomer and the expert, the casual viewer and the aficionado alike. Your family and friends will enjoy the two hours of exciting shots, behind the scenes action, and bonus interviews with winners, designers, vendors, and judges.

PDXBOT.04 is available from Lemon Studios for only $24.95, plus shipping and handling.

For further information, please contact:

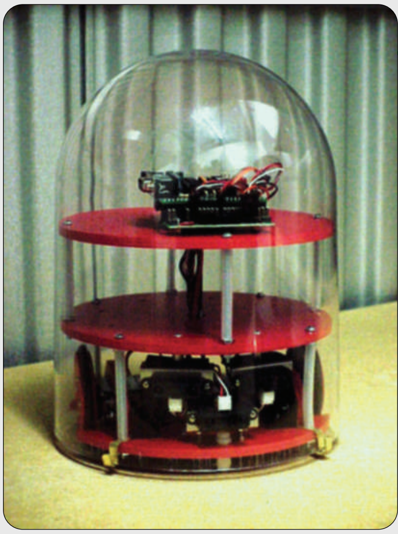
Lemon Studios

1424 SE 162
Portland, OR 97233
Tel: **503 • 761 • 0964**
Email: **info@lemonstudios.com**
Website: **www.lemonstudios.com**

Circle #132 on the Reader Service Card.

MENAGERIE

Send us a high-res picture of your robot with a few descriptive sentences and we'll make you famous. Well, mostly. menagerie@servomagazine.com



Domebot

Chuck Hellebuyck, Commerce, MI

The Domebot is a multilevel, expandable, autonomous robot built from off-the-shelf components. It has an acrylic dome that makes it an R2-D2 style robot and protects the electronics, sensors, and any add-on items — such as a wireless camera — but still allows the sensors and cameras to operate unobstructed. It can be built for under $150.00 and is a great platform for schools, hobbyists, and even home use as a roving household droid.

Domebot has a two servo differential drive with front and rear skids to balance the chassis. It also has three front infrared detection sensors that can sense objects to the left, right, and center of its forward path. It has individual wheel sensors mounted at the servos to monitor individual drive wheel rotation for accurate turns and stall sensing. A low profile line detection sensor is also mounted under the main platform.

An Ultimate OEM module that accepts programming in Atom Basic, PicBasic Pro, C, or Assembly controls the Domebot. The Ultimate OEM plugs into an OEM Robotics Docking Station that makes connecting sensors, batteries, and servos as simple as plugging in a wall socket. Expansion is made easy with multiple predrilled holes for sensor mounting. The base and multiple levels are built from expanded PVC, so modifications to the chassis are simple.

www.elproducts.com

Hacked Roomba

Chris Waters and Nick Kelsey, San Jose, CA

The Roomba is an autonomous robotic vacuum cleaner that uses a variety of sensors to navigate itself around your living room. Robotic vacuuming is a more difficult task than it might seem at first, as both navigation and the physical design are problematic. The vacuuming nozzle requires a lot of sucking power and there is a rotating brush that creates a lot of friction against carpet. This means that the Roomba packs a hefty battery and very powerful motors.

Nick Kelsey (an avid Tivo hacker) and I wanted to see what we could do with a Roomba, so we broke out the screwdrivers and took a couple of Roombas apart to see how they work. The main objective was to see if we could control how the Roomba navigates; therefore, we needed a way to change the control algorithms. The CPU in the Roomba is an 8051 derivative, but — unfortunately — it is not in-field upgradeable, so we needed another way to run our own code. We decided to remove the CPU and use a Parallax Javelin Stamp module instead. The Javelin Stamp module is perfect for controlling small robots and we figured that using Java would be great if we wanted to do more complicated things with our new robot, like maze solving.

We spent many hours with a multi-meter working out how the pins on the CPU mapped to the Roomba's sensors and motors, in addition to what the signals looked like when the sensors were activated. Then, the 8051 CPU was carefully desoldered and fly leads were connected to each of the pads left on the Roomba's motherboard. These fly leads could then be connected to the breadboard on the Javelin Stamp Demo Board and we were ready to control the Roomba ourselves. With a lot of trial and error, we wrote a Java program that generates the appropriate PWM waveform to drive the two independent wheel motors on the Roomba and we were away.

The flat top and high motor torque make the modified Roomba a wonderful platform for future robotics experiments.

Our resident expert on all things robotic is merely an Email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

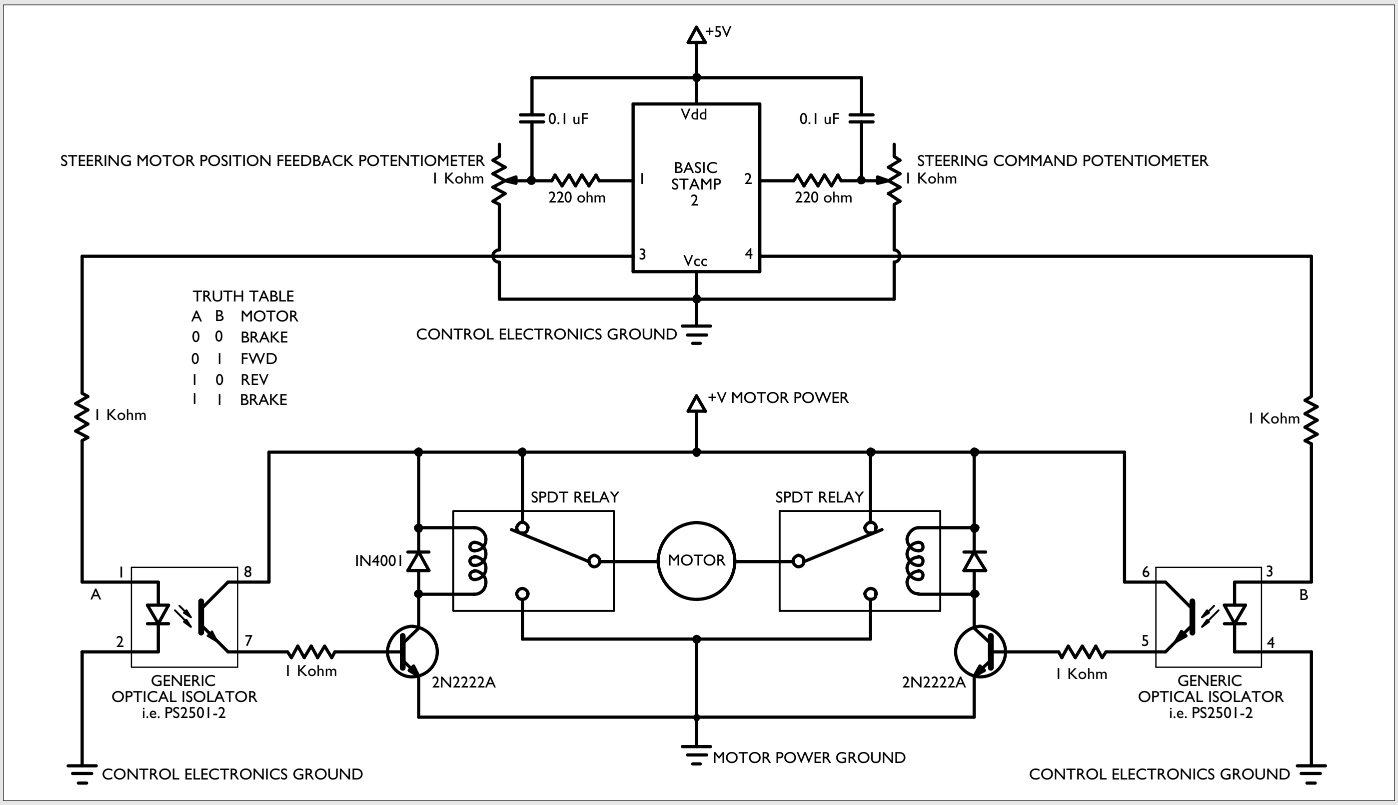
by
Pete Miles

Q. I am trying to improve upon the steering capabilities of our new double stroller. I would like to add "power steering" to it and, eventually, a power drivetrain. My basic idea (as I am new to electronics and robotics) is that I will have to hook up a pretty beefy DC motor for the steering. Someone at Parallax recommended that I incorporate the Motor Mind C that will control up to two 24 V motors at 4 amps, continuous, for the steering along with the Motor Mind C Carrier Board to simplify the process.

Well, the process couldn't get simple enough for me because I have no idea how to do any of this. Anyway, I would like to activate this motor using two pressure sensitive pushbuttons — one for each hand. Push the left-side button and the wheels turn to the left and so on. The harder you press, the quicker the wheels turn. Of course, the wheels should return to the default, straight ahead position when pressure on either button is removed, just like an R/C car, but with a straight DC rather than servo motor.

I think the stroller should also be able to roll freely, unless the motor is activated — much like an electric bike. Am I getting in over my head here? Conversely, if I did decide to have the stroller only powered, I would like the motor to be

Figure 1. Relay-based steering servo motor controller.



able to act as a brake, also.

Is the Motor Mind C an actual motor controller, along the lines of the much, much more expensive AmpFlow controllers? How hard would it be for a novice to learn how to program the code for such a project?

I'm sure I will be a regular customer and visitor to your site. If you could offer any advice, that would be great.

— Mike Harman
via Internet

A. No, you are not over your head on this type of a project. You just need to break it down into several smaller projects. This project has two major tasks: powered steering and powered drive. Strollers typically have four wheels, where the front wheels are mounted on casters and the rear wheels are fixed. Let's begin with the powered steering aspect of this project. You mentioned using pressure sensitive switches to control how fast the wheels turn. I am going to assume that what you meant here is that the harder you press the button, the greater the turning angle the front wheels will go through. Since you wanted the wheels to return to their straight ahead position when the pushbuttons are released, then you are going to need some sort of feedback to sense the position of the wheels/motors.

For the DC motor, I would recommend a low speed gear motor with an output shaft speed that doesn't move any faster than about 60 RPM. Automotive windshield wiper or window opening motors work well here and they typically run at 12 V. The motor can be connected directly to the front wheel's caster axle (which will require two motors) or the front wheels can be configured like a rack and pinion system, typically used in cars. In that case, only one steering motor would be needed.

A potentiometer can be used as a simple feedback sensor for monitoring the steering wheel's position. This should be mounted directly to the steering system in such a way as to be able to measure the full range of the steering mechanism while maximizing the full range of motion of the sensor. Mounting directly to the caster's axle is one suggestion.

Probably the hardest part of this project is installing the pressure sensitive pushbuttons. The pressure sensitive sensors that you are talking about here are known as Force Sensors. These sensors are generally very expensive and designed more for industrial applications. One low cost option is the IESP-12 force sensor, made by CUI Stack, Inc. (**www.cui.com**), and available from Digi-Key (**www.digikey.com**) and HVW Technologies (**www.hvwtech.com**). This sensor acts like a resistor, where the resistance changes as the force on the pushbutton increases. Another option is to use a rotary or linear potentiometer, a lever, and a spring attached to the lever to move the lever back to the home position when the pressure is released from the lever.

Figure 1 shows an electrical schematic on how to implement a simple steering control circuit using a potentiometer as a steering feedback sensor, a potentiometer for the pushbutton control, and a couple of SPDT relays for the motor direction control. A BASIC Stamp 2 is used as the microcontroller in this application. The capacitors wired to the potentiometer are there for the RCTIME function of the BASIC Stamp. They can be removed from the circuit if you are using a microcontroller that has an analog-to-digital converter. The PS2501-2 is an optical isolator to protect the microcontroller from voltage surges when driving the motor.

The program shown in Listing 1 is a simple example program for controlling the steering direction of the motor using a single input potentiometer as the pushbutton sensor.

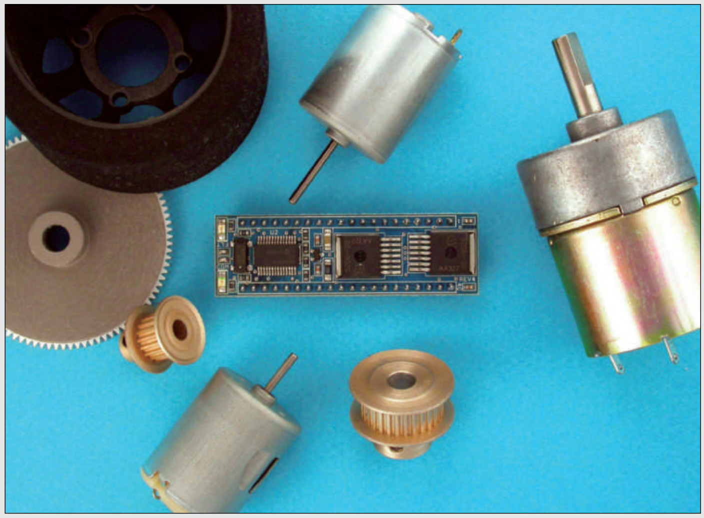


Figure 2. The Motor Mind C from Solutions Cubed.

All this program does is compare the position of the steering feedback potentiometer position and the pushbutton position. When the difference between the two readings is greater than the dead band, then the logic will tell the motor to move until the difference is less than the dead band. The dead band is needed to reduce the amount of motor oscillating about the end position. This value can be increased or decreased, depending on how much sensitivity you want in your system.

Listing 1

```
'{$STAMP BS2}
'{$PBASIC 2.5}

' Sample program that demonstrates how to use a regular DC gearmotor,
' a position feedback potentiometer, and a steering position command
' potentiometer to perform as a servo motor.

Motor_Pos       VAR  Word        ' Steering motor position feedback
Steering_cmd    VAR  Word        ' Commanded steering motor position
Dead_Band       VAR  Word        ' Dead band size variable
tmp             VAR  Word        ' temp variable

' Initialize steering motor to its current brake position
LOW 3                         ' H-Bridge relay control A
LOW 4                         ' H-Bridge relay control B
Dead_Band = 20                ' Set dead band width to 20

Main:    ' Main loop
  GOSUB Read_Motor_Position
  GOSUB Read_Steering_Command
  IF Motor_Pos > Steering_cmd THEN
    tmp = Motor_Pos - Steering_cmd
    IF tmp > Dead_Band THEN
      GOSUB Turn_Left
    ELSE
      GOSUB Stop_Motor
    ENDIF
  ENDIF
  IF Steering_cmd > Motor_Pos THEN
    tmp = Steering_cmd - Motor_Pos
    IF tmp > Dead_Band THEN
      GOSUB Turn_Right
    ELSE
      GOSUB Stop_Motor
    ENDIF
  ENDIF
GOTO main

Read_Motor_Position:       ' Read current steering motor position
  HIGH 1
  PAUSE 1
  RCTIME 1, 1, Motor_Pos
  RETURN

Read_Steering_Command:    ' Read current steering commanded position
  HIGH 2
  PAUSE 2
  RCTIME 2, 1, Steering_cmd
  RETURN

Stop_Motor:                ' Stop the steering motor
  LOW 3
  LOW 4
  RETURN

Turn_Left:                 ' Turn the steering motor to the left
  LOW 3
  HIGH 4
  RETURN

Turn_Right:                ' Turn the steering motor to the right
  HIGH 3
  LOW 4
  RETURN
```

Depending on the potentiometer values and the range of motion of the steering mechanism and input potentiometers, the measurements for the potentiometers may need to be scaled up or down. This can be done by multiplying the values by some constant. The constants will depend on the actual system built. A second input potentiometer for controlling the motor position in the other direction can be added to the circuit. As for the logic, averaging the two input positions together will work for control of the motor position. In essence, this is a simple circuit for making a servo motor using a regular DC gear motor. The second part of this project is the drive system. Since the front wheels are being used for steering, only a single motor is needed to drive the rear wheels. Because strollers are fairly light, the same windshield wiper/window opening motors should be sufficient to drive the stroller. A heavy duty motor choice would be to use a wheel chair motor from National Power Chair (**www.npcrobotics.com**).

The advantage of these motors is that they are low speed and have a lot of torque for driving a stroller. The drawback to these motors is that they really don't spin freely, due to their gear boxes. So, the requirement that the stroller move freely when the motors are not in use will require some sort of a mechanism to disengage the motor. Another option here is to use a plain DC motor with a 3/8 inch or 1/2 inch diameter shaft and press the shaft against the wheel of the stroller or use a large disk on the wheel's axle and use a flat belt between the disk and motor shaft. There is more chance for the motor to slip with this case, but — when the power is removed from the motor — it will freely spin. If you want the motors to act like a brake when using the stroller in the powered mode, then you should use a gear motor.

The Motor Mind C (**www.solutionscubed.com**) is a true motor controller and is very similar to the AmpFlow (**www.ampflow.com**) motor controller. Both of these motor controllers will work in this application. They both can control one or two different motors. They both can accept RS-232 serial communication from a microcontroller, a 0-5 V analog input signal from a potentiometer, or a standard 1-2 ms R/C style pulse width for controlling motor speed. The main difference between the two controllers is how much current they can handle. The Motor Mind C has a maximum current draw of 4 amps, whereas the AmpFlow motor controller has a maximum current draw of 160 amps. Which motor controller you use really comes down to how much current your drive motor requires.

The Motor Mind C Carrier Board is a convenient board that holds the 40-pin Motor Mind C module

and a BASIC Stamp 2 module. The board has an RS-232 port along with screw terminals for the motors and motor batteries. The RS-232 port is for programming the BASIC Stamp that is used to control the Motor Mind C. There is a 16-pin header for the Stamp to connect to different sensors. This system can be used to control both the steering and drive motors. The Motor Mind C can replace the SPDT relays for driving the steering motor.

The AmpFlow motor controller has another advantage over the Motor Mind C; it has both velocity and position control for the two motors it controls. Thus, this controller can be wired directly to a drive motor and steering motor with an encoder and no special electronics will be needed to control the motors (other than the pressure sensitive pushbuttons).

Solutions Cubed also has two other products that will work well at directly controlling the position and velocity of the steering motor. They are called the Mini PID and ICON PID motor controllers and they have 3.75 and 12 amp continuous current handling capabilities, respectively. Another option is to look at the motor cotrollers from Vantec (**www.vantec.com**). They offer both velocity and position controllers. They have a unique controller called the RBSA Bully servo amplifier that works just like the one shown here.

As for the input sensor for setting the speed of the stroller, you could use a similar pushbutton type of control like one being used for the steering or you can use a spring controlled lever like those seen on lawn mowers. Here, a potentiometer and a spring connected to the lever are used for the speed control sensor. You have to hold it down in order for the motor to run. You should set up about half of the range of the lever to keep the motor in the stopped position and the final half of the range of motion would be the variable motion that is used to control the speed of the motor. This way, if you let go of the lever, then the stroller will immediately stop.

This will be a fun project. All you need to do is break it down into several smaller parts, get each part working by itself, and then combine the parts together. I hope this gives you enough information to get started. **SV**

TETSUJIN TECH

Here is a sneak peek at the teams currently building for *SERVO Magazine's* powered exoskeletal weight lifting competition.

Xela

New Canaan, CT

Team Leader: Alex Sulkowski, financial services consultant

Background: I am an army of one on this project. My full time job is consulting to credit card companies and engineering is my hobby. I spend my free time incorporating electronics into mechanical devices. Recently, I have been building R/C cars, making them autonomous via GPS and computers.

Motivation: The degree of difficulty was appropriate and the timing was right because I had been feeling the urge to enter an engineering competition for a while. In college, I built many devices — including a 500 lb remote operated tank, a wheel chair that climbed up stairs, a skateboard with a gas engine (long before today's scooters were around), and touch-sensitive skin for a robot.

Strategy: I will build a suit that utilizes the operator's strengths, along with the strengths of the suit, while minimizing their respective weaknesses. My suit is very flexible — with little resistance in all of the operator's joints — so the operator can walk by providing the necessary coordinated motion. When the weight is being lifted, many of the suit's joints reach the limit of their range of motion and bear the weight of the load — making only a few pneumatic pistons necessary.

Largest Obstacle: The laws of physics. More seriously, time is the major obstacle. With a full time job, a wife, and two kids — and doing this project solo — there is never enough time.

Academic Focus: I attended Yale, officially in economics, but most of my time was spent in mechanical and electrical engineering. I also attended Johns Hopkins, studying bio-medical engineering (low cost ultrasound imaging).

Construction Materials: The frame will be made of steel.

Power Source: Compressed air for the pneumatics.

Estimated Cost: $1,000.00

Contact: ASulkowski@argusinformation.com

Mechanicus

Austin, TX

Team Leader: Jascha Little
Team Engineer: Scott Little

Background: My dad and I are both mechanical engineers with experience in system design, microprocessor controls, and automation. We are the creators of The Judge — a combat robot seen on the TV show *BattleBots* and winner of four trophies.

Motivation: I've been a big fan of science fiction since I was a kid. The powered exosuits of *Starship Troopers* have been a dream of mine for a long time. Plus, I love a new challenge.

Strategy: Maximum everything! The suit is designed around a 3,000 lb maximum lift capacity, yielding a 100% safety factor. Every joint is independent of the others; I didn't want to put a big control computer on it.

Largest Obstacle: Time is my constant nemesis. I was also surprised by the inefficiency of hydraulics. I originally thought I'd run the whole thing from a giant accumulator, but discovered the resistance of the servo valves was much higher than I expected. This could become a test bed for more efficient hydraulic technology.

Academic Focus: I attended UT Austin in the mechanical engineering program, though I did take a lot of electrical engineering classes because I found them interesting.

Construction Materials: I am using regular old steel because it's cheap. I'm also trying an alloy called AR400 for the legs. It is commonly used in construction equipment, like in bulldozer scoops. It is barely machinable.

Power Source: Compressed air drives the hydraulics. It is both light and power dense.

Estimated Cost: $10,000.00

Contact: halo@austin.rr.com

COMPETITOR PROFILE

For more information on the event, visit
www.servomagazine.com/tetsujin2004/

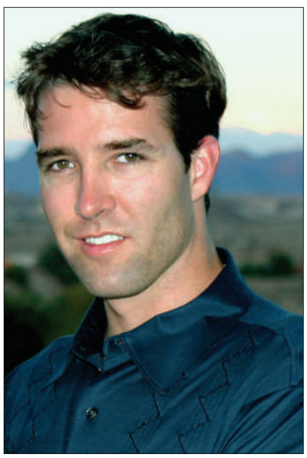
Team Raptor

El Segundo, CA

Team Leader: Chuck Pitzer, design and simulation specialist

Background: I just like to create things and do nice work. I like the process of development and improvement. I consult on various mechanical development projects, typically utilizing Pro/ENGINEER — the design and development software from PTC.

Motivation: It's right up my alley and fits my skill set nicely. I used to build combat robots, but I'm burned out on them. I am intrigued by this event; it's a competition that is meaningful. There are so many possibilities for this technology — from the military, to helping the disabled, to applications in industry.



Strategy: My plan is to develop a solid design using off-the-shelf materials for this first year, while minimizing the complexity of the control system. I plan to lift the weight to the maximum height, without focusing either on lift time or overall system weight. If I can't lift the full 720 kg this year, no sweat!

Largest Obstacle: The biggest obstacle I expect will be to stay balanced while lifting the weight. Also, of course, there is time. It's an optimization project. It's not impossible to do, but can it be done well?

Academic Focus: I attended the University of Florida, majoring in mechanical engineering.

Construction Materials: I am using tubular mild steel because it is inexpensive and simple to work with.

Power Source: The power plant for the suit will be an electro-hydraulic setup. When you're talking about these types of forces, hydraulics are the safest way to go.
Estimated Cost: $2,000.00

Contact: www.pitzerconsulting.com

Robytes

by Dave Calkins

Another month, another collection of robot trivia to amuse your coworkers and annoy your pub-mates. Surely there are more fun stories out there. Got a good story on robots? Email me: news@robotics-society.org If you'd like to get even more robot news delivered to your in-box (no spam, just robo-news) drop a line: subscribe@robotics-society.org

— David Calkins

Robot Dinosaurs Attack!

Photo courtesy of Kokora Dinosaurs.

Stephen Spielberg, eat your heart out! Why bother making dinosaurs with computer-generated graphics that only live on screen when you can have tangible dinosaurs that move around and scare the bejesus out of your kids?

Well, that's the plan at California's Los Angeles Zoo. Rather than having the normal, non-moving fiberglass dinos, zoo Director John Lewis hired Kokoro Dinosaurs to make animated beasties that move their heads, swing their tails around, and snarl at you.

Kokoro — a Woodland Hills, CA company — makes robot dinos of all sizes and shapes, from swimmers like Elasmosaurus and Mosasaurus up to more famous land-dwelling dinos like the Raptor, Tyrannosaurus Rex, and Triceratops.

Though they have a metallic skeleton, their skins are lifelike and they do a good job of educating kids — not only about the terrible lizards, but also the terrible cost of extinction. The goal of the LA Zoo's exhibit is not only to entertain, but to warn us about the dangers of extinction that many animals face right now.

Although I, for one, wouldn't complain if mosquitoes went the way of the Dodo ...

Here, There Be Dragons!

Photo courtesy of the US Marine Corps.

Not to be outdone by the Army and their pack-bots, the Marine Corps Warfighting Laboratory's Reconnaissance Surveillance and Target Acquisition Technology Section and Carnegie-Mellon have developed their own combat robot, named the Dragon Runner. In typical mil-speak, the Marines list these exciting facts:

"The Dragon Runner unmanned ground vehicle weighs about 16 pounds and fits inside a Marine's Modular, Lightweight, Load-carrying Equipment (MOLLE) patrol pack. It is designed to provide Marine small units with increased situational awareness and tactical force-protection capability within the urban environment. The Marine Corps Warfighting Laboratory tested the Dragon Runner vehicle at Quantico, [VA]."

The lightweight bot is no lightweight when it comes to punishment, though. It can survive 14-foot drops and be flung through windows, over walls, and down staircases — all without damage.

If only my martini-swigging friends were so durable.

BugBots Get Smaller

Photo courtesy of Epson Corporation.

Some days, we all want to be the fly on the wall. Well, in that effort, Epson has further refined its tiny microFR flying chopper. The old version had wires to an external battery, so it could not travel very far. The new model has placed the batteries onboard, refined the gyro-sensor to one-fifth of its predecessor's weight, boosted power by 30%, and managed to increase its attitude control mechanism for even better flights. However, they didn't stop there. They've added a wireless camera that can transmit images via a Bluetooth controller back to the operator!

Finally, something to spy on my editors and find out what they do on those three-hour lunches ...

Robot Attacks Wedding Party!

What happens when you're President of the Robotics Society of America, Head Judge of the Robot Fighting League and BattleBots, Co-Chief of Security for Survival Research Labs, and founder of ROBOlympics — and yet — through it all — you still

Photo courtesy of Scott Beale.

decide to marry a human? Why, the robots revolt, of course!

Half-way through the ceremony — to the surprise of the audience — a huge robot came trundling down the aisle, past the superstars of the robotics world. It ran ahead of Mark Setrakian and Peter Abrahamson, zoomed by Steve and Nora Judd of Tentacle Robotics, crunched past Zander Rose of Inertia Labs, ignored BattleBots co-founder Greg Munson, and zipped by *SERVO* editor Dan Danknick.

The 480-pound robot didn't even glance at Karen Marcelo, Violet Blue, and Kimric Smythe of SRL or Charlie Gadeken of QBox and Power Tool Drag Racing fame as it raced toward the wedding couple.

It lumbered up to the bride and groom and raised three of its six legs six-feet into the air ...

The audience gasped!

Will this be the end of our hero and heroine?

Will they be crushed under the might of Mechadon?!?

No! Our heroes soothed the beast and the audience was relieved to find that it had only brought the hand-wrought wedding rings to the happy couple. They took the rings from the beast and felt honored to have had such an esteemed bot make their day perfect.

Yup, I got married to my best friend and we had robots in our wedding party (after spending our wedding morning together, disassembling, moving, and re-assembling a flame-thrower ...) Hey — it's my column (and it's a social column at that), so I get to wax poetic when I want to. Thanks, Mark and Pete.

The Most Awesome Robot — Ever

If you're a typical robot nerd, your morning routine is less than enjoyable and usually centers around coffee and

Photo courtesy of Nicholas Blye.

some minimal form of nourishment. Well, I have stumbled upon the deeply guarded solution to your pre-10 AM woes: the Breakfast Express. Originally announced in 1994 by Welbilt for about $400.00, the Bex will gladly offer room service-like treatment right in your kitchen (basement, lab, workshop, etc.) Simply insert a couple of eggs, some bread, and coffee and set the desired time.

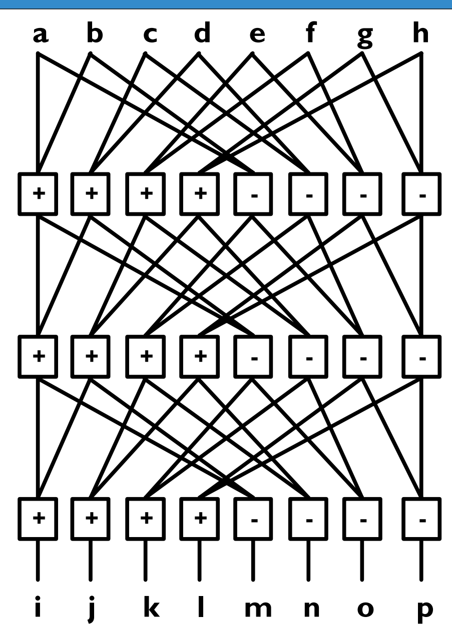
Bing! Stainless steel blades lop the bottoms off the eggs. They cook them in a non-stick pan while toasting and brewing occurs simultaneously. You're ready to just roll out of bed (cot, floor, welding table) and bon appetite! SV

Optimization Using a GENETIC ALGORITHM

by Sean O'Connor

You may be familiar with function minimization and maximization using the method of Hook and Jeeves or the downhill simplex method of Nelder and Mead. Simulated annealing methods are sometimes good for more highly dimensional problems with lots of local minima or maxima to contend with, although selecting suitable "cooling" rates is problematic to the point of being an art. Another alternative is to use genetic algorithms. They can be very effective, but often have a large number of tricky parameters to choose from. The algorithm I present here requires a straightforward population size parameter and a noise parameter and that is it. General guidelines about appropriate values can be given. Another advantage of this algorithm is that it is very simple to implement, requiring only about 100 lines of code in most high level languages, which is comparable to the simplest method.

Figure 1. The Hadamard Transform. In this case, the number of elements is $8=2^3$, so we need three add subtract layers. We can divide by the square root of 2 at each add and subtract, ensuring that the vector length remains constant throughout the transform or we can scale the result afterward.



Gaussian Noise and Searching Out From the Centroid

A well-known way to generate random numbers with a Gaussian (or Normal) distribution is to add together several random numbers from a uniform distribution, such as those from a typical **rnd()** function that returns a number between 0 and 1.

If, instead of adding all the random numbers together, you add half of them and subtract the other half, the resulting Gaussian distribution will have 0 mean.

Table 1. The output values of the Hadamard Transform.

```
p=(a-b-c+d-e+f+g-h)*c
o=(a+b-c-d-e-f+g+h)*c
n=(a-b+c-d-e+f-g+h)*c
m=(a+b+c+d-e-f-g-h)*c
l=(a-b-c+d+e-f-g+h)*c
k=(a+b-c-d+e+f-g-h)*c
j=(a-b+c-d+e-f+g-h)*c
i=(a+b+c+d+e+f+g+h)*c
```

In fact, adding and subtracting equal numbers of random numbers from almost any distribution (for example, a random function that returns a 1 10% of the time and a 0 90% of the time) will give you a Gaussian distribution, as long as you use enough of them.

The reason for using Gaussian noise in genetic algorithms is that adding the right amounts to a search point in each of its dimensions (e.g., the x and y dimensions in a plane) results in an effective search policy.

A genetic algorithm has a population of vectors, each representing a point in one or more dimensions, depending on the function you are trying to minimize or maximize. The centroid of a population of vectors is the mean or average value in each of its dimensions.

In the algorithm I present, you construct a new population of vectors by adding Gaussian noise to this centroid. The correct amount of noise to add in each of the dimensions is, in fact, determined by the variance of the current population in that dimension, multiplied by a scaling factor of slightly over 1.

Once you have your new population, it is time for the law of the jungle to be applied — namely, survival of the fittest. You combine the two populations and sort them according to fitness, keeping the best and saying goodbye to the others. Over several generations, there is an almost inevitable increase in fitness.

Implementation

If you randomly permute an array

Table 2. A Java implementation of a genetic algorithm (continued on page 64).

```
package abc;
import java.util.Random;

public class Genetic {
// Global variables.
  static int popSize;
  static double[][] population;
  static Random rnd;

// Information about the function to minimize
  static int dimensions=2;
// lowest and highest input values allowed
  static double elemMin=-2.048;
  static double elemMax=2.048;

// Set population size and noise over-scaling
  static int lnPop=6;          // population size =2^5
  static double scale=1.6;     // noise scaling factor
  static int generations=2000;

// Program starting point.
  public static void main(String[] args) {
    double[] best;
    initialize();
    for(int i=0;i<generations;i=i+1){ minimize(); }
    best=getBest();
    System.out.println(); System.out.println();
    System.out.println(“Lowest Cost:”+function(best));
    System.out.println(“At:”);
    for(int dim=0;dim<dimensions;dim=dim+1){
      System.out.println(best[dim]);
    }
  }

  public static void initialize(){
    rnd=new Random();  // create rnd number generator
    popSize=1<<lnPop;  // shift left by lnPop
// In the population array you add an extra dimension
// for the cost of each individual and also provide
// storage for the next generation in the upper half
// of the population array.

    population=new double[dimensions+1][popSize*2];

// randomly fill current and next generations with 0..1
    for(int dim=1;dim<dimensions+1;dim=dim+1){
      for(int elem=0;elem<popSize*2;elem=elem+1){
        population[dim][elem]= elemMin+rnd.nextDouble()*
                                (elemMax-elemMin);
      }
    }
    cost(0);  // fill in the cost dimension
    sort();   // sort the population by cost
  }

// call this repeatedly and then call getResult to get
// the best current result. rnd.nextInt(n) returns a
// random number between 0 and n-1 inclusive.
  public static void minimize(){
    double[] work=new double[popSize];
```

of numbers that are not all equal, the resulting sequence looks like it has been obtained from some random distribution (though likely not a uniform one). Adding and subtracting equal numbers of the numbers in the array will produce a Gaussian distribution with a variance that is directly related to the variance of the original numbers in the array. This works in general, but it is, of course, easy to defeat by having only one non-zero array element, for example.

If you consider the diagram of the Hadamard transform in Figure 1, particularly its coefficients (listed in Table 1), you can see that each of the outputs of the Hadamard transform — except one (representing the mean of the inputs) — is just some pattern of equal numbers of adds and subtracts. Hence, if we transform a randomly permuted array of numbers, each of the non-mean outputs will be a sample from a Gaussian distribution.

To deal with the case of only one non-zero input, all you have to do is randomly permute the output of the first transform and transform it again. What is happening is that the first permute shifts the non-zero value around, the first transform maps it to a regular looking pattern, the second permute mangles that pattern, and the second transform turns that mangled pattern into noise. There are other cases you can figure out, but they are not important for this application.

If you arrange for the second permute to leave the value that contains information about the mean alone, then the second transform will have the mean of the original data imposed on all its outputs. In addition, it will have 0 mean Gaussian noise added to all its outputs with the same

Table 2 continued ...

```
  double temp;
    int swapPos;
    for(int dim=1;dim<dimensions+1;dim=dim+1){
      for(int i=0;i<popSize;i=i+1){
        //copy dimension data (lower half)
        work[i]=population[dim][i];
      }

      // randomly permute work array
      for(int i=0;i<popSize;i=i+1){
          swapPos=rnd.nextInt(popSize);
          temp=work[i];
          work[i]=work[swapPos];
          work[swapPos]=temp;
      }
      work=transform(work);          // Hadamard
Transform of work array
      for(int i=1;i<popSize;i=i+1){// multiply all
except work[0] by noise scale
         work[i]=work[i]*scale;
      }
      for(int i=1;i<popSize;i=i+1){// randomly permute
work array except work[0]
          swapPos=rnd.nextInt(popSize-1)+1;
          temp=work[i];
          work[i]=work[swapPos];
          work[swapPos]=temp;
      }

      // Hadamard Transform work array
      work=transform(work);

      // copy up to next generation region
      for(int i=0;i<popSize;i=i+1){
         temp=work[i];                 // checking bounds
         if(temp>elemMax) temp=elemMax;
         if(temp<elemMin) temp=elemMin;
         population[dim][i+popSize]=temp;
      }
    }
    cost(popSize);    // fill in the cost dimension
    sort();           // shift down the fittest
  }

// Insertion sort. Sort together the current and next
// populations so that the best of both go to the lower
// half of the population array and the worst into the
// upper half ready to be overwritten.

  public static void sort(){
    int j;
    double temp;
    for(int i=1;i<popSize*2;i=i+1){
      j=i-1;
      while(j>=0 && population[0][j] >
                    population[0][j+1]){
         for(int dim=0;dim<dimensions+1;dim=dim+1){
            temp=population[dim][j];
            population[dim][j]=population[dim][j+1];
            population[dim][j+1]=temp;
         }
         j=j-1;
       }
     }
   }

// fill in the cost dimension (ie population[0][...]).
  public static void cost(int from){
    double[] vector=new double[dimensions];
    for(int indv=from;indv<popSize*2;indv=indv+1){
      for(int dim=0;dim<dimensions;dim=dim+1){
        vector[dim]=population[dim+1][indv];
      }
      population[0][indv]=function(vector);
    }
  }

// Hadamard transform
  public static double[] transform(double[] input){
    double[] a=new double[popSize];
    double[] b=new double[popSize];
    double[] temp;
    double cVal=1/Math.sqrt(2);      // adjusting factor
    for(int i=0;i<popSize;i=i+1){    // copy the input
      a[i]=input[i];
    }
    for(int times=0;times<lnPop;times=times+1){
      for(int pos=0;pos<popSize;pos=pos+2){
        b[pos/2]=(a[pos]+a[pos+1])*cVal;
        b[(popSize+pos)/2]=(a[pos]-a[pos+1])*cVal;
      }
      temp=a;           //swap arrays a and b
      a=b;
      b=temp;
    }
    return a;
  }

// returns best vector so far.
  public static double[] getBest(){
    double[] result=new double[dimensions];
    for(int i=0;i<dimensions;i=i+1){
      result[i]=population[i+1][0];
    }
    return result;
  }

// This is the function we are trying to find a
// minimizing set of values for.
// Rosenbrock's saddle function min 0 at (1,1)
// limits -2.048 to 2.048.
  public static double function(double[] vector){
    double result=(vector[0]*vector[0]-vector[1]);
    result=100*result*result;
    result=result+(1-vector[0])*(1-vector[0]);
    return result;
   }
}
```

variance as that of the original data.

I mentioned that it was necessary to use slightly over scaled Gaussian noise for the genetic algorithm. The appropriate place to do that is to multiply all the outputs of the first transform — except the value representing the mean — by the required scaling factor.

Creating a new population is now pretty easy. You apply the above scheme to each dimension in the current population to create the corresponding dimension in the new population (see Figure 2).

I have provided a code listing that

can be seen in Table 2 (and downloaded from the *SERVO Magazine* website, **www.servomagazine.com**). The code is in Java, but I have deliberately written it so that it should be very easy to convert to other programming languages.

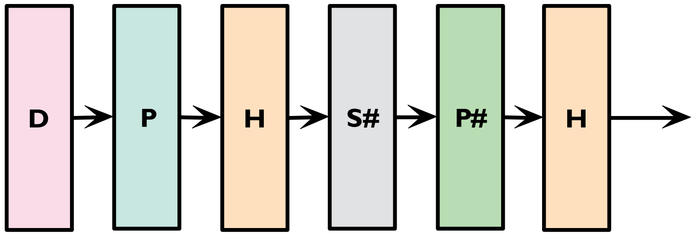
General Guidelines

The scaling factor should generally be between 1.1 and 2.0. At values close to 1, you will get a rapid convergence to a local minimum/maximum. For problems in two, three, or four dimensions, you can use a population size of 64 or 128 with a scaling factor of 1.5 or 1.6. For more highly dimensional problems, you can use a population size of 128 to 512 and scaling factor of between 1.1 and 1.2.

For really difficult problems, where finding a global minimum or maximum is hopeless, but where there are many good local minima or maxima, you can actually use a scaling factor of less than 1 — say between .9 and .99 — together with a large population size of between 256 and 1,024.

Conclusion

This algorithm falls midway between random hill climbing — where you evaluate points entirely at random — and more complex genetic algorithms with lots of parameters to choose from. The number of function evaluations required can be quite high, but — in exchange for that — you get an effective search through the problem space that makes few assumptions about the function you are trying to optimize. This is ideal for more complex problems. The shear simplicity of the algorithm is also rather appealing. **SV**



D = A dimension in the current population.

H = Hadamard transform

P = Random permutation

P# = Random permutation leaving the sum*c value fixed.

S# = Apply scaling factor leaving the sum*c value unchanged.

Figure 2. Preparing a dimension for the next population.

Simulated Annealing

If you half melt a plastic bowl, form all sorts of depressions and hills in it, then let it cool, the result could be some sort of a representation of mathematical function. To find the minimum — or lowest point — of this function, you can scatter a few grains of sand into the bowl and hope they provide an indication of the lowest point. Unfortunately, most of the grains of sand will be trapped in local depressions in the bowl and will not fall to the absolute lowest part.

Borrowing an idea from physics, where high temperature means high random motion of atoms, molecules and particles at the microscopic scale and low temperature means low random motion. You can "heat" things up for the sand grains in the bowl by shaking it. With enough random motion, the sand grains will be able to jump out of the local depressions they are stuck in.

If you gradually "cool" the system down by reducing the amount of shaking, then, eventually, all the sand grains will fall to the lowest part of the bowl. Computer scientists have mimicked this approach in order to solve difficult problems and have termed it "simulated annealing."

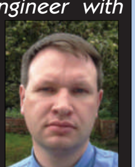
The term annealing comes from metallurgy, where cooling a metal at different rates gives it different properties (hardness, ductility, etc.). Steel, for example, is made up of iron and carbon atoms. Now, the iron atoms would prefer to be in large iron crystals and the carbon atoms would prefer to be in carbon rich areas. However, at the very high temperatures where steel is made, the iron and carbon atoms have such large amounts of random motion that the two are thoroughly mixed.

Cooling steel quickly is the same as shaking our plastic bowl vigorously and then suddenly stopping. The sand grains in the bowl and carbon atoms in the steel will be distributed all over the place in small clumps. Cooling steel slowly, on the other hand, allows the gradual separation of the iron atoms into large crystals and the carbon atoms into large, carbon-rich sheets, as both types of atoms fall into their preferred minimum energy situation. By cooling the steel slowly, you are — in effect — solving a very complex problem in energy minimization.

If you are having trouble understanding how the program in the main text works, you can think of it as being similar to simulated annealing. The noise is equivalent to random motion and this is automatically adjusted in a downward trend as the problem is solved.

About the Author

Sean O'Connor is an engineer with Marco Beverage Systems. More information about the Hadamard transform is available at: ***http://uk.groups.yahoo.com/group/htapps/***




4 out of 5
mice read
SERVO ...
LYNX
Shouldn't you?
SUBSCRIBE NOW!
12 issues for $24.95
www.servomagazine.com or
call toll free 1-800-783-4624
Buy Robotic Arm at www.lynxmotion.com

DARPA Grand Challenge *Spawns* SRS/SERVO Magazine Robo-Magellan — Part 3

by Michael Miller

This is part three of a series of articles discussing my entry for the SRS/*SERVO Magazine* Robo-Magellan contest. The goal of the Robo-Magellan contest is to find and touch an orange cone that is placed somewhere in the Seattle Center. The robot must be fully autonomous and the only information that the robot will have about the location of the cone is its latitude and longitude coordinates. Details of the contest can be found at the Seattle Robotics Society's Robothon webpage (**www.robothon.org**).

This article will focus on the microcontroller selection, the core tasking system, sensor management, and lastly, the conversion of latitude and longitude coordinate systems into a local coordinate system

The Microcontroller

When I started designing my entry bot, I had several choices that I could make for the hardware brains. My background is in Win32 development, so it would have been an easy fit for me to select a small computer board that could run one of the Windows variants, like Windows NT 4.0, Windows XP, Windows CE, or even something new, like Windows XP Embedded. Lately, though, I have been using several of the Atmel line of AVR chips for various small projects and the more powerful of them was also up for consideration.

The Windows platforms are all powerful operating systems that I am comfortable using to their full advantages, but I had to consider the hardware they ran on and how they fit into my solution. I needed something compact and not too power hungry to fit on my rather small chassis with its limited battery carrying abilities. This restricted me to the embedded versions. I hunted for some small boards that supported it and considered the costs.

I found that one of the smaller and lower cost solutions was to use a PocketPC PDA. It would come with its own battery and user interface. Used ones could be found for a real bargain, but it would require an external hardware support to drive all lower level I/O I needed, like the IR range sensors and servos. This would require that I either purchase such support or build my own. I could build one using one of the Atmel AVR microcontrollers easily enough, so I investigated that direction.

While researching the design of having all I/O driven off this external microcontroller and the serial I/O to the PocketPC, I started to consider that the PocketPC might be overkill for project needs and that an Atmel Mega128 could do both jobs. I had already built a framework of code for the Atmel AVR line that could handle many of the I/O demands and timing solutions (more details on those systems later) and, from previous projects, I had keypad and LCD support.

So, I ran a few CPU utilizations tests running all the inputs and outputs I thought I would need, which included many PWM out, several timers, several PWM in, all ADC channels, and some LCD output. I clocked my Mega128 at 16.384 MHz. I found that all the overhead of my framework and the I/O took less than 2%

of the CPU cycles, leaving me with plenty of cycles to do other work.

I concluded that I should be able to do all I want on this project with the Atmel Mega128 for the main brain, as far as computation cycles go. Then, however, I had to consider whether it could handle all the I/O requirements. The current list of requirements was as follows:

5 ADC pins — IR proximity sensors
2 ADC pins — compass directional sensor
2 Interrupt pins (PWM in) — accelerometer used for the head auto level
2 Servo PWM out pins — servos used for head auto level
1 Servo PWM out pin — motor speed controller
2 Servo PWM out pins — servos used for steering
1 Servo PWM out pin — servo used for the CMUcam panning
2 In pins — bumper
1 Interrupt pin — failsafe trigger
4 Interrupt pins — wheel encoders
1 Serial — CMUcam
1 Serial — GPS
1 Serial — PDA communications for waypoint download
7 output pins — LCD control using four-bit interface
1 SPI — QMatrix QT60040 keypad
1 Output pin — piezo buzzer

Several problems were apparent. The Mega128 only has two hardware supported serial lines, but I needed three. It also only had eight external interrupts, which — after the second serial port — used two of the same pins; the PWM in routine would use another two and the failsafe trigger used another one. That only left three available for the wheel encoders. The built-in support for PWM could just do six outputs, but the resolution for hobby servo use was poor and I might have the need for more servos in the future.

I could handle the third serial port requirement. Since the PDA connection would only be needed before the start, I could set up a method that disconnected the GPS from a serial port and connect the PDA temporarily. Once the waypoints were downloaded, the connection to the GPS could be reconnected. I considered this problem solved.

Solving the problem of lack of more external interrupts required more thought. I could move the wheel encoder work into another microcontroller, but it would then need to know the active motor direction to correctly accumulate the counts. This would require timely communications from the main CPU handling the motor direction. Furthermore, it would require communications to get those counts back to the main CPU for use.

It isn't a bad solution, but it does require another microcontroller with at least four external interrupts and the design and coding of multi-processor communications protocol. I could also just poll the failsafe trigger. It would free up the last needed interrupt, but I would risk adding some latency to when the bot will react to the use of the failsafe. I was confident that I could handle it within 200 ms and felt this was an adequate safety margin. This was the direction I went with.

I handled the servo PWM by creating my own routine. One of the timers supported the ability to have three compare interrupts. This would allow me to create a timer for the high pulse (about 4 ms total), set the pins high for a set of 3 PWM out, then set these three compare values to a group of three servos output values. On the interrupts of those compares, I would set the pins associated with those servo outs to low. On the overflow of that timer, I would reset to the next three servo readings and repeat five times for a total of 20 ms. This had very little overhead and provided a great many hobby servo outputs.

With all this thought out, I decided that there was little risk in going with the Mega128 as the main CPU. For the final brain hardware solution for the bot, I wanted to design

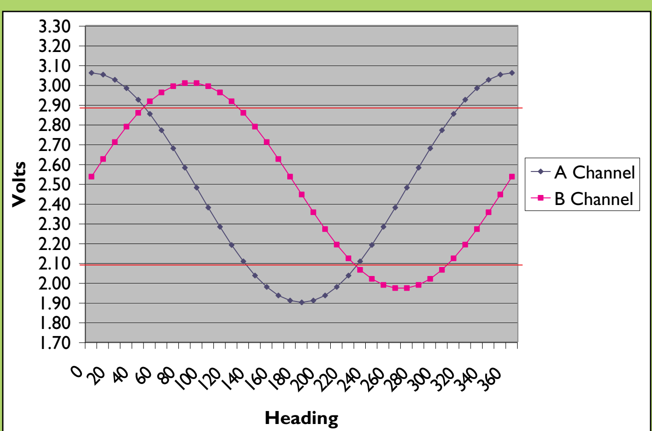
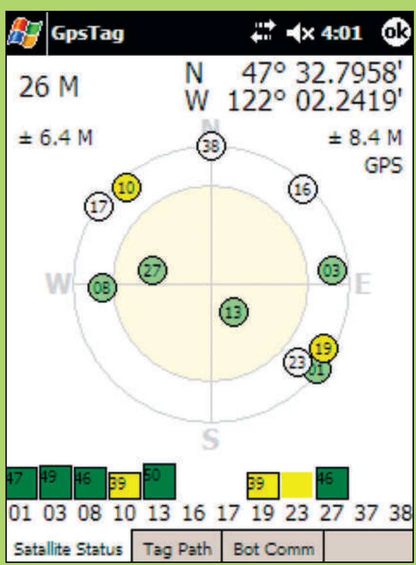


Figure 1. This graph plots the voltage from each of the compass output channels when the heading changes. There is a sine/cosine relationship between the output channels, but their respective ranges differ. Note the high and low value where the two signals cross, as marked with the horizontal red line. Between these ranges, the curves are mostly linear.

Figure 2a. The PocketPC application has a screen for graphing satellite locations and signal strengths, very similar to most graphic GPS units. I include the current latitude and longitude and the error readings on this screen. This is very useful for knowing how good the signal is at that location.



my own circuit board that supported the keypad and LCD with a piezo buzzer on it. It would have headers to a second circuit board that would handle I/O interconnections to the sensors and other devices. This would allow me to reuse the primary board design for different projects while customizing the second board for whatever project was on hand. For now, I am continuing with the experimenter's board I have as I develop the software and systems and I will continue working on the final board layout.

The Operating System

We are all familiar with some of the very well-known operating systems, like MS-DOS, Windows, Macintosh OS, and Unix. The definition of an operating system is basically a framework of software that hosts applications and abstracts details of the hardware to provide common services needed from all applications.

In the microcontroller arena, custom OS are designed and built all the time — often without the creator really thinking about it. They build frameworks that can be reused from project to project that handle task switching, serial I/O, PWM out, PWM in, keyboard inputs, and user outputs, like LCDs.

There are a few off-the-shelf, real time operating systems available for the Atmel AVR microcontrollers. One of the more popular ones is AvrX, which provides preemptive task switching, along with support for timers, semaphores, and a message queue — all the core features you will need for multi-tasking operations. Another popular OS is FreeRTOS, which has similar capabilities to AvrX.

In general, due to the needs of diverse microcontroller projects, OSs often do not abstract the hardware and solely provide core task switching needs. They require that the project creator provide the rest of the support. Lucky for anyone developing for the Atmel AVR, there are plenty of examples to start with that are already written in both assembly and C. A good place to look is **www.AvrFreaks.com**

For some of my previous microcontroller projects, my goals were to design a user interface through simple text-based LCD and keypad entry — something similar to what a cell phone would have a few years ago. To do this, I had built a non-preemptive tasking system with timers and message queue support. It made it much easier to add new pieces and run multiple tasks in a simple, but effective, manor. I am reusing this body of code.

So, why did I create a non-preemptive tasking system, rather than a preemptive one? It's a complicated issue and the answer lies in what is right for me and my projects. Mostly, it came down to memory use. Many of the I/O operations are already handled by interrupts, so timely management of them was not an issue. It also came down to the UI being the biggest part of previous projects.

In a non-preemptive tasking system, the task continues with full control of the CPU until it gives it up or some normal interrupt routine borrows it for a small portion of time. A preemptive tasking system will — at a consistent time interval — interrupt the running task mid-operation, store away its state, reset another task's state, and then let the new task run until the next time period. The ability to handle external events in a consistent and short time period is what will define an OS as being real time, not just because it's preemptive; don't get the two definitions mixed up.

The code for storing and restoring the state of task requires that CPU cycles save all the registers and swap the stack pointer. Remember that this store/restore code happens on a consistent and often short time period. None of this is specifically needed in a non-preemptive system. I say specifically, as the same register save/restore actions happen already whenever you make a function call within your code.

In the case of a function call, they are optimized to what is used by that routine. In the case of the preemptive task switch, it must blindly store/restore all of them. When the non-preemptive task does give up its processing time, it is returning from a function call and, thus, similar code is invoked, although much less often and always optimized to what would be used within it.

A preemptive system inherently has the requirement for a separate stack per task that is large enough for the needs of that task. Go beyond a few tasks and this will quickly eat at what little RAM these controllers have. The size of the stacks has to be tuned to the task so as not to waste memory with underused stack space. A non-preemptive system has no need for separate stacks, as it shares a standard one.

A non-preemptive system has one very important requirement, though. That requirement is the need to keep all code separated into small pieces to minimize the period between tasks. In the preemptive system, this is controlled by the task switching period, but, here, it must be handled by each task, which must keep its code length down or separate one logical block of code into many smaller pieces with a state machine. It relies on a message system to manage task actions. One poorly written routine taking too long during its "reign" can add significantly to delays and glitches in other tasks. Further, writing code this way often isn't very intuitive for most developers. However, if you have ever done much UI coding, you will find that it is not that different.

Support Routines

Along with the functions to manage the tasks, I created a few functions to abstract other common things I needed. These can be grouped into the categories of LCD, keypad, timers, PWM out, PWM in, ADC in, EEPROM, and serial. Some are purely function calls, while others have a mix of functions you call and task messages that are received.

The task code is tied closely with the message handling code. It manages the message queue and details of calling the message tasks when a message is available to it. Other systems will call into this system to use the message features.

The LCD code currently handles all the issues by managing a four-bit parallel interface to the text LCD. This is a common interface that many text-based LCDs support. The supported

functions are pretty straightforward like Initialize, DisplayOn, WriteChar, WriteString, etc.

The timers code currently uses one hardware timer to manage many timer requests. The functions allow the registration for a timer message with a 1 ms resolution and the ability to accurately pause using this timer.

The PWM servo out code handles the details of keeping the pulses going to the servo. This is not a general purpose PWM out system, as it was designed to be optimal for the standard hobby servo pulse definition of a 20 ms cycle with a pulse that varies in width around the 1.5 ms center. It has the resolution of 2,048 steps for a normal pulse range and 4,096 for the extended range, which I found hard to achieve with the built-in PWM out support. It can also support up to 24 outputs if output pins are available; although, as currently coded, the pin assignment is hard-coded.

The PWM in code handles the details of constantly monitoring the pulses on any of four external interrupt pins. It is very much a general use set of routines and can be used for not only the input of a standard hobby servo routine, but also — in my case — for reading the accelerometer. It returns a filtered value (currently an average of the last 10 pulses) or the last complete reading. The values are defined by the times the pulse was high and the time of the following low. Currently, the pin assignment is hard coded to the channel.

The ADC in code handles the details of the built-in ADC. It supports full control of the channel for single pin reading (differential readings are currently not supported). It uses the sample complete interrupt to run a series of samples in order and then notifies the task by a message that the sample was completed. This allows for readings of a series of similar sensors (like the compass) to be sampled close together for better value matching.

The serial code handles the details of the built-in USARTs. It supports an interrupt-driven input and output and I added buffering to both.

Lastly, the EEPROM code wraps the intricacies of reading and writing to the onboard EEPROM.

Example

An example of a simple program that will create a single task follows. It defines the task procedure (TestTaskProc) and it shows the creation of the task as modal. This specific creation function will not return until the TestTaskProc ends. The task creates a timer when it gets initialized. The timer is defined to happen every 100 ms. On the first occurrence of that timer, it will destroy that timer and close the task.

```
#define c_idClockTimer 1
uint8_t TestTaskProc( Message* pMsg ){
  Bool fReturn = false;
  switch (pMsg->id) {
    case MSG_Initialize:
      CreateTimer(pMsg->idMsgTask, c_idClockTimer, 100);
      fReturn = true;
      break;
    case MSG_Timer:
      if (pMsg->Param8 == c_idClockTimer){
        DestroyTimer( pMsg->idMsgTask, c_idClockTimer);
        MsgTaskCloseModal( pMsg->idMsgTask, 0 );
      }
      break;

    case MSG_Destroy:
      fReturn = true;
      break;
  }
  return fReturn;
}

int main (void ) {

    MessageQueueInitialize();
    SysTimerInitialize();

    // start primary task
    MsgTaskCreateModal( TestTaskProc, 0 );
}
```

This is not an attempt to document or instruct the full use of my OS, but is meant to give you a glimpse into the calls that I will use, so that latter examples will have more context. You will find similar functions in almost all embedded OSs. I don't consider my framework to be done, as I still have far too many hard-coded mappings for this to be general enough, but it is good enough for my ongoing use and I will continue to evolve it with any future projects.

Analog-to-Digital Conversion

Since I was using Sharp IR ranging sensors for forward object and ground detection, I needed to be able to read the signals they provided into something useful that the software could use. The Atmel AVR microcontrollers — like many microcontrollers today — have a single analog-to-digital converter onboard. It supports the ability to select which channel or pin is being sampled by this converter, but only one at a time. Further, the AVR supports the ability to start the sample and then have an interrupt happen when the sample is done, allowing the code to continue doing other things.

My functions that wrap the ADC work by setting up the initial channel to sample, as defined by the caller's list of channels to sample, then — on each sample complete interrupt — it will store away the value from that sample and set up the next one to sample automatically. Once it has reached the end of the list of channels, it will then send a message to the requested task that the samples are complete. An example task procedure follows. This task is set up to sample the two channels 0 and 1, using the max number of bits 10, using the external voltage reference, and restart the sample every 40 ms. The sample's values would be referenced and acted upon inside the sample complete case, but, here, I just assign them to variables.

The values' returns are based upon the voltage reference defined and the quality of sample (in bits) that was requested to be taken. In the case above, if the external reference voltage was 5 V, then the 10-bit sample would return values

in the range of 0 through 1,023, representing 0 V through 5 V. For the Sharp IR range sensors, this is fine, but not optimal. The specs on them show that the analog output will never go above 3.3 V. If the reference voltage is maintained at 3.3 V, this will greatly improve the resolution of the values, as the 0 through 1,023 will now represent 0 V through 3.3 V.

However, the Mega128 only has one external voltage reference. So, if I had mixed requirements — with some being 5 V and some being 3.3 V — this will not work. Even though it does support an internal voltage reference of both 5 V and 2.5 V and you can switch between the internal references on the fly, neither of these can be used if the external reference is connected to another power source. The only other need I had for ADC was the compass. It also output an analog signal that was below 3.3 V and would benefit greatly, as well. So, I could continue with the external 3.3 V reference.

The compass analog signal was more complex to handle than IR range values. IR range values were used with some simple trigger's values to know if something was closer or if the ground had fallen away. The compass signals — when graphed by heading — will show a sine and cosine relationship between the two signal lines (see Figure 1). Note that the two signals do not have the same range. The key to the conversion of these two signals lies in the use of the fairly straight diagonal portions of each signal where they cross.

So, when the compass is pointed toward 90°, signal A will output about 2.5 V, while signal B will output about 3 V; signal A is on the straight downward slope, while B is near its peak of the curve. If the values where the two curves cross is known (and updated, as these values will roam with temperature), then a few simple comparisons against these known values will tell us which general quadrant we are pointed toward and which signal to use to calculate the heading angle. In my example above, the fact that signal B is above the crossing points tells us to use signal A's value and that we are in the NE quadrant.

Here is the code I used to calculate the approximate heading:

```
…
case MSG_AdcSampleComplete:
  if (pMsg->Param8 == c_idCompassSample) {
    uint16_t SampleA = Adc10BitSample( 0 );
    uint16_t SampleB = Adc10BitSample( 1 );
    // direction in 1/10th degrees
    uint16_t heading = 0;
    uint16_t sample = 0;
    // if equal, keep updating the cross points
    if (SampleA == SampleB) {
        // find min and max
        if (SampleA > 512)
            s_CompassCross.max = SampleA;
        else
            s_CompassCross.min = SampleA;
        s_CompassCross.range = s_CompassCross.max -
s_CompassCross.min;
    }
    if (SampleB >= s_CompassCross.max) {
      // 45 - 135
      heading = 450;
      // inverted
      sample = range - (SampleA - s_CompassCross.min);
    }
    else if (SampleA <= s_CompassCross.min) {
      // 135 - 225
      heading = 1350;
      // inverted
      sample = range - (SampleB - s_CompassCross.min);
    }
    else if (SampleB <= s_CompassCross.min) {
      // 225 - 315
      heading = 2250;
      sample = SampleA - s_CompassCross.min;
    }
    else if (SampleA >= s_CompassCross.max) {
      // 315 - 45
      heading = 3150;
      sample = SampleB - s_CompassCross.min;
    }
    uint32_t temp = 900 * (uint32_t)(sample) /
```

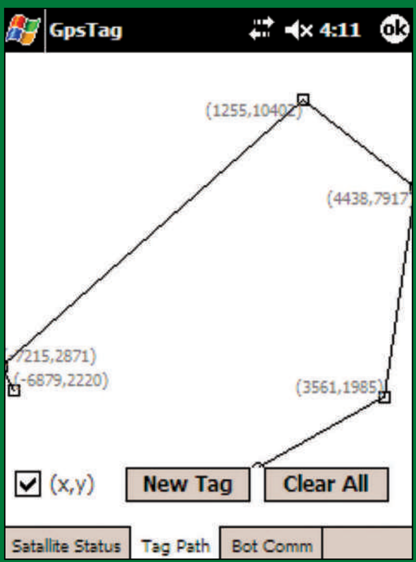


Figure 2b. This screen of the PocketPC application displays the waypoints defined and allows me to add and edit them. The last location is constantly updated to the location of the GPS unit.

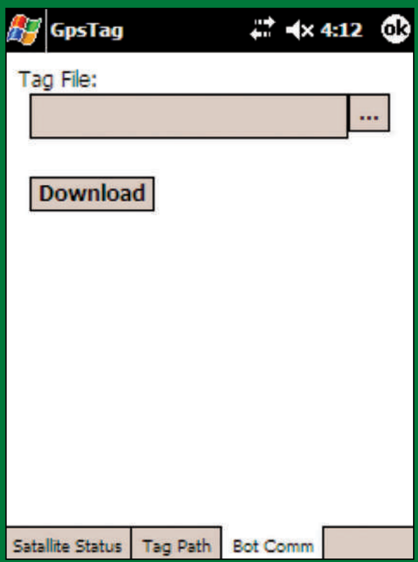


Figure 2c. This screen of the PocketPC application allows me to set the file that the waypoints are stored in the PocketPC and allows me to start the download into the bot.

```
(uint32_t)( s_CompassCross.range);
    heading += (uint16_t)temp;
    if (heading >= 3600)
            heading -= 3600;
  }
  break;
…
```

It is a little more complex than just this, though. What happens when both values drift greater than the high cross point or lower than the low cross point, but are not equal? I currently handle this by just averaging the two samples with the previous cross point and use the result as the new cross point.

So, for each 90°, there is about a 0.87 V spread. With a 3.3 V reference using 10 bits, each value change will represent about 0.003 V. This equates out to 0.333 of a degree per sample value. Although my calculations show we have better than a single degree of accuracy, this does not take into account signal drift, interference issues, and the fact that our diagonal line is not straight, but does have some curve. So, I never rely on anything other than 1 degree and round my heading appropriately.

Keeping a Level Head

One of the key features I designed on the bot was the auto leveling head. This would provide the required level surface for the compass. The design uses an accelerometer chip that is used to provide a tilt angle. Each axis for the sensor is aligned to one of the two servos so that readings for each channel can be directly supplied to each servo. This accelerometer chip outputs two PWM signals that define the tilt angle. The values are not discrete measurements that will tell you exactly what angle the head is at. They are neither related nor are they stable over temperature changes. So, I designed the system to be able to resample the values for leveling before each use and hope that the drift over 15 minutes would be minimal. So far, this has been true.

How do you apply the readings to the servos to adjust the head to level, though? This same problem has been solved many times before through the use of a PID equation. I will not go into detail on how it works or even how to tune one, as this has been covered many times by other authors, but I will explain how I did it in my code:

```
PwmInValue pwmValue;
int16_t PwmReading;

// get new value
PwmInGetReading(iChannel, &pwmValue);
PwmReading = PwmFromT0T1(pwmValue.T0,
                         pwmValue.T1);
// the PID
int16_t error = s_aPwmPid[iChannel].Desired -
                   PwmReading;
int16_t derivative = error -
        s_aPwmPid[iChannel].ErrorPrevious;
int16_t pwmAction = s_PidGain.ProportionalGain *
  error + s_PidGain.IntegralGain *
s_aPwmPid[iChannel].ErrorIntegral / 4 +
  s_PidGain.DerivativeGain * derivative;
                s_aPwmPid[iChannel].ErrorIntegral +=
error;
s_aPwmPid[iChannel].ErrorPrevious = error;

//convert pwmAction into a servo control value
int16_t ServoValue = PwmGetServo( iChannel );
ServoValue -= pwmAction / 64;

// limit the servo values to the normal range
if (ServoValue > c_PwmMax)
  ServoValue = c_PwmMax;
else if (ServoValue < c_PwmMin)
  ServoValue = c_PwmMin;

// set the servo value
PwmSetServo( iChannel, ServoValue );
```

There were a couple of key things that needed to be taken care of to stabilize the system beyond the PID tuning. One was that the accelerometer signal isn't very stable and the second was that hobby servos can't react very quickly due to their communications protocol.

One of the key things that I found while creating this solution was that the servo change would not affect the next tilt reading in time. To explain this issue, you need to understand how the PWM signal for servos works. It provides a pulse every 20 ms that the servo reads and then tries to act upon. If I am changing the servo value faster than every 20 ms, it is useless, as the servo may not have even gotten the next signal, let alone any internal latency it has. So, the PID cycle had to happen at a slower rate than the servo cycle to give it time to react and allow the next readings to be affected. Therefore, I set the PID cycle to 40 ms.

The second issue was that I needed to filter and smooth out the readings from the accelerometer. The accelerometer allows its PWM out to be tuned by the use of an external resistor. This allows the specific application to get the cycle it needs. In my case, I wanted to get enough samples to filter it for every cycle of the PID. I felt that 10 samples would be good and I set the resistor to give me a 4 ms cycle.

With these improvements and a lot of manual PID tuning, the head unit would maintain a pretty stable and level base while standing still. Currently, it will tilt forward when the bot accelerates. It doesn't tilt enough to be a concern, but I am working on removing that acceleration from the readings. Things can always be improved.

Coordinate Systems

Latitude and longitude are the standard coordinate system to use for locations on Earth. They represent the angle from the equator and the angle from an origin longitude to the location. It is a nice system when navigating a sphere (ellipsoid, actually), but the math involved when trying to calculate distances between two points is complex. On the

Robothon website (see Links sidebar), the rules actually list the equations. Below, I have inserted some code that implements those equations; it is in C# written for the PocketPC using Microsoft Visual Studio .NET:

```
public double DistanceBetweenPoints( GpsWaypoint pt1,
GpsWaypoint pt2, double Radius ) {
  double dLon = pt1.Longitude - pt2.Longitude;
  double dLat = pt1.Latitude - pt2.Latitude;
  double a =
   Math.Pow(Math.Sin(AngleConvert.Radians(dLat/2.0)),
   2.0) + Math.Cos(AngleConvert.Radians(pt1.Latitude)) *
   Math.Cos(AngleConvert.Radians(pt2.Latitude)) *
   Math.Pow(Math.Sin(AngleConvert.Radians(dLon/2)),2.0);

   return Radius * 2.0 * Math.Atan2( Math.Sqrt( a ),
   Math.Sqrt( 1.0 - a ));
}
```

To improve the accuracy of this calculation, the Earth's approximate radius near the locations should used. Since my bot isn't that interested in latitude and longitude, I decided to use a local Cartesian coordinate system. I convert latitude and longitude to my coordinate system when it is sampled from the GPS and do all other math in my coordinate system. Since my bot is not planning on traveling large distances (hundreds of km), its thinking the world is flat in my local space is not an issue.

The course is defined to have the starting cone and the ending cone within 300 feet of each other. This is a little over 91 meters. If I take the starting cone as my origin and consider the above distance, I could use a 16-bit signed integer to represent a coordinate value and keep a cm resolution. This would give a ±327 meter range, which is well outside any possible path that could lead to the cone. I felt this was adequate.

To convert latitude and longitude to the local coordinate system, I needed to know the conversion scale near the location for each axis. This is easily calculated by plugging in the current location and the current location with a small offset in one axis into the above distance equations. This will give you the distance on that axis for the offset defined. Here's an example C# code:

```
/ calculate 1 degree shift in lat
GpsWaypoint Shift = new GpsWaypoint( location );
if (90.0 - Shift.Latitude > 1.0)
  Shift.Latitude += 1.0;
else
  Shift.Latitude -= 1.0;

LatToKmScale = DistanceBetweenPoints( location, Shift,
radius );

// calculate 1 degree shift in lon
Shift = new GpsWaypoint( location );
if (180.0 - Shift.Longitude > 1.0)
  Shift.Longitude += 1.0;
else
  Shift.Longitude -= 1.0;

LonToKmScale = DistanceBetweenPoints( location, Shift,
radius );
```

With these scale values, you can now convert latitude and longitude into the local coordinate system with the following equation:

```
public const int c_LocalPerKm = 100000; // cm in km
// we want coordinates in cm
x = (int)((origin.Longitude - location.Longitude) *
LonToKmScale * c_LocalPerKm);
y = (int)((origin.Latitude - location.Latitude) *
LatToKmScale * c_LocalPerKm);
```

My solution is to walk the course and create waypoints before the start of the competition. Since the rules state that no robot can be brought with me, I needed a lightweight method to create the waypoints that used an auxiliary GPS unit. I decided to use my PocketPC, tied to a serial GPS unit that was of the same make and model that I was going to use on my bot.

I used the .NET Compact Framework and C# to develop the software that would get information from the GPS and show the status of the lock (see Figure 2). Furthermore, I added the ability to "tag" waypoint locations and show a map in local coordinates of those points. Lastly, it would download the original latitude and longitude, along with the local coordinates and conversion scale to the bot.

Next Time

I didn't get to cover as much as I wanted to this month. The event is coming up quickly and the work needed to complete the bot is stacking up.

If you would like to download more examples of the control software I've written, visit the *SERVO Magazine* website at **www.servomagazine.com**

In this article, I mentioned the CPU selection and OS, along with how sensor information is handled. I talked about the conversion to my local coordinate system and the use of a PDA to tag waypoints.

In the next article, I hope to cover my dead reckoning methods and include some testing observations. Wish me luck! SV

Links

www.Atmel.com — *makers of the Mega128 microcontroller*

www.AvrFreaks.net — *info on the Atmel Avr chip line*

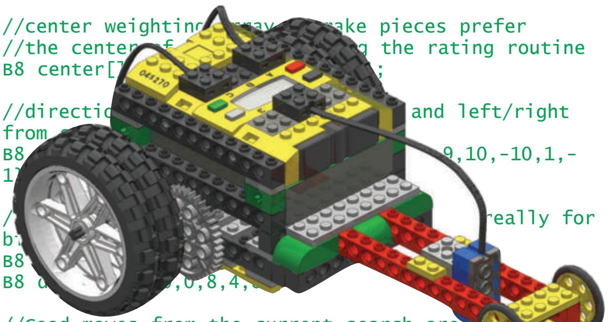
www.barello.net/avrx — *AvrX real time kernel*

www.FreeRTOS.org — *FreeRTOS site*

msdn.microsoft.com/mobility — *Microsoft Embedded developers' site*

gcc.gnu.org — *GCC home, a free C complier for the Atmel AVR line*

```
// castling bonuses
B8 castleRates[]={-40,-35,-30,0,5};

//center weighting array to make pieces prefer
//the center of the board during the rating routine
B8 center[
```



```
//Good moves from the current search are
this array
//so we can recognize them while searching and make
sure they are tested first
```

A bi-monthly column just for kids!

LESSONS FROM THE LABORATORY

— PART 5 —

Getting Your Gear On

by James Isom

If you think about it, almost every machine with a motor has some gears alongside, helping out. Gears — when combined with an axle — are one of six simple machines that help provide ***mechanical advantage*** to a machine. What's mechanical advantage, you say? It's the ability for a machine to do more work, so you don't have to. Gears are often used in a machine to change the speed, amount of power, or direction of motion or to transfer energy from one place to another.

In the LEGO world, we identify a gear by the number of teeth it has. The gears in Figure 1 are 24-tooth gears. The number of teeth a gear has is often shortened to the number of teeth, plus the letter "T," as in a "24T" gear.

Gear teeth mesh together to transfer energy from one place to another. Two or more gears in combination are called a ***gear train***. The gear connected to the motor is called the ***input gear*** because it is providing the initial input of energy to the meshing gears. The gear being moved is referred to as the ***output gear***. For every tooth that moves on the input gear, one moves on the output gear. This characteristic can be used to improve the mechanical advantage of a machine.

In the example shown in Figure 1, the gears are of the same size and move tooth for tooth in opposite directions. For every rotation of the "input" gear, the "output" gear rotates once, as well. This is called a 1 to 1 gear ratio and is written as "1:1" with a colon representing the word "to." In this case, the gear train isn't providing a mechanical advantage, except for the fact that it is reversing the direction of rotation. What we're putting in is about what we are getting out, with the exception of a little lost to friction.

Different gear sizes can be combined to create different gear ratios and can, thereby, be used to create different degrees of mechanical advantage. For instance, on each side of the example robot we've been using — "Eddie*" — we have an eight-tooth input gear (8T) meshing with a 40-tooth

Figure 1. Gear train.

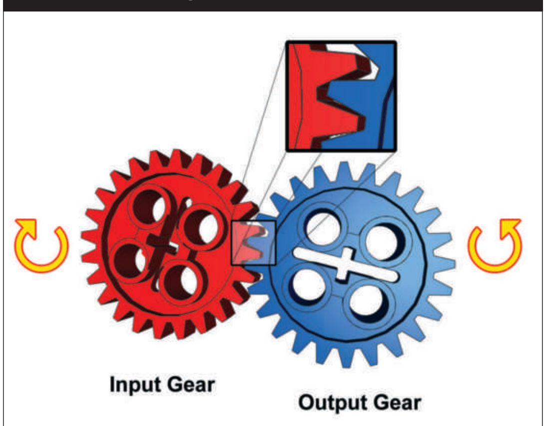


Figure 2. Figuring gear ratios.

8:40

$$\frac{8}{8} \quad \frac{40}{8} = \frac{1}{1} \quad \frac{5}{1}$$

5:1

* Building instructions for "Eddie" can be found in the February 2004 issue of *SERVO Magazine*.

(40T) output gear on each axle. To figure out the gear ratio, we need to find out how many times the input gear must rotate in order to make the output gear rotate once.

To do this, take the sizes of the two gears and make them a ratio of 8:40. To figure out the exact gear ratio, we need to reduce the ratio by the largest number that can be divided into both 8 and 40 equally — which, in this case, would be 8. This gives us a 5:1 gear ratio, which tells us that, every five turns of the 8T gear, the 40T gear rotates once.

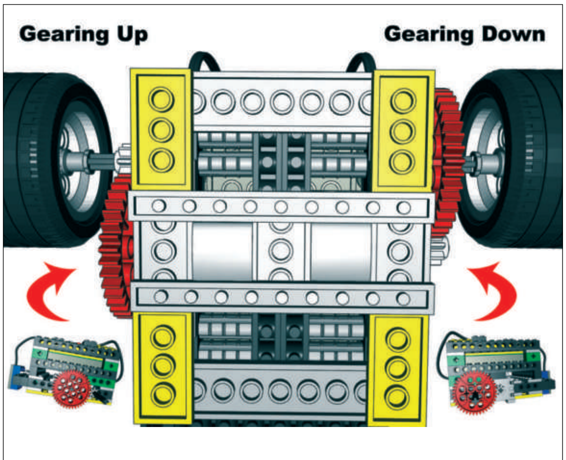


Figure 3. Eddie's gear switch.

To really understand how gears can help us, we need to learn about ***torque*** (rhymes with "spork"). Torque is the twisting force that we use every time we twist the top off a jar of pickles, open a door, or turn a nut on a bolt. I'm not going to go into all the details of how torque works here; I'll leave that to your future physics teacher or a quick Internet search. Let's just say it's a force that produces rotational motion.

Take a moment and set up this quick experiment to help illustrate the relationship between gear ratios and torque. Remove the front wheels from "Eddie" for a minute and switch the gears on one side so that there is an eight-tooth input gear on one motor and a 40-tooth input gear on the other motor. Do the same for each rear axle so that there is the exact opposite gear ratio on each side: 5:1 on one and 1:5 on the other. When a small input gear is driving a larger output gear, it is called ***gearing down***. A large input gear driving a smaller output gear is called ***gearing up***.

Program your robot so that the motors run in the forward direction or simply press the "View" and "Run" buttons at the same time to turn on all three power ports in the forward direction. While holding your robot in your hand, run the program; when the wheels start spinning, you will notice that one wheel is obviously moving much

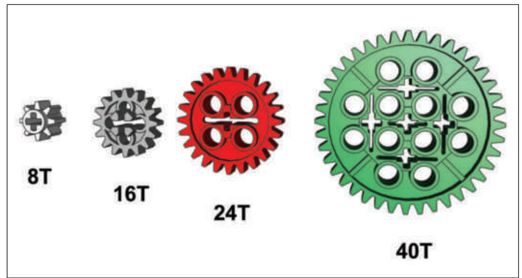


Figure 4. Spur gear "family photo."

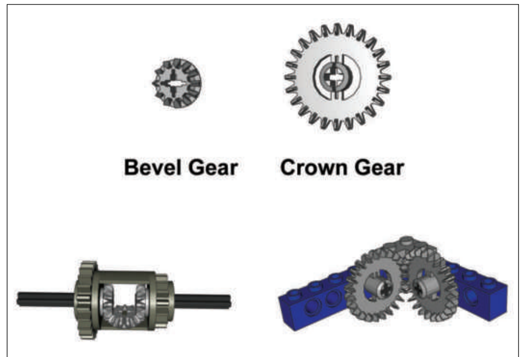
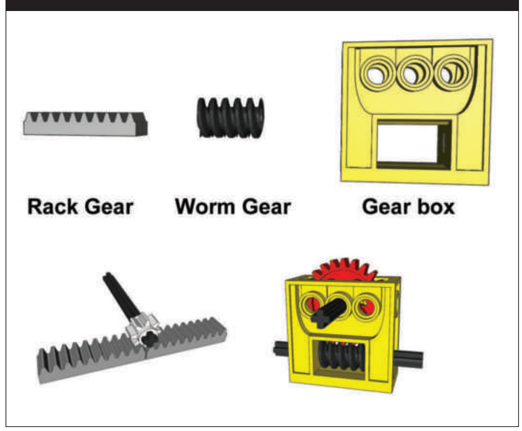


Figure 5. Bevel and crown gear "family photo."

Figure 6. Worm and rack gear "family photo."



GLOSSARY

Input Gear: Sometimes referred to as the drive gear. It is the gear that is usually attached to a motor that provides power to a machine.

Output Gear: Sometimes referred to as the driven or follower gear. An output gear is turned by an input or idler gear.

Gear Train: Two or more gears working together to transmit power from one place to another.

Gearing Up: The output gear is moving faster than the input gear, causing an increase in speed, but also an equal decrease of torque.

Gearing Down: The output gear is moving slower than the input gear, causing an increase of torque, but also an equal decrease in speed.

Idler Gear: A gear in the middle of two other gears, usually used to reverse the axis of rotation of the output gear.

Mechanical Advantage: The output force produced by a machine divided by the input force applied to a machine.

Torque: Simply put, torque is a twisting force. It is the product of a force multiplied by the distance of that force from the center of the turning radius (or fulcrum). Torque = the force x the radius.

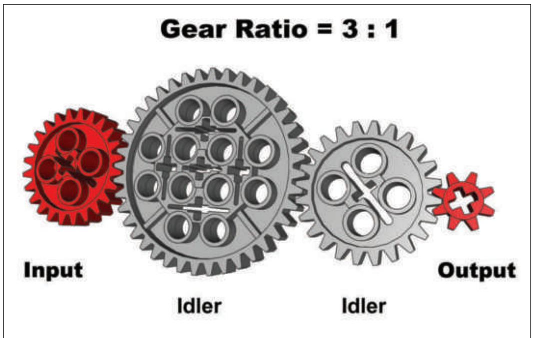


Figure 8. Gear train with idler gears.

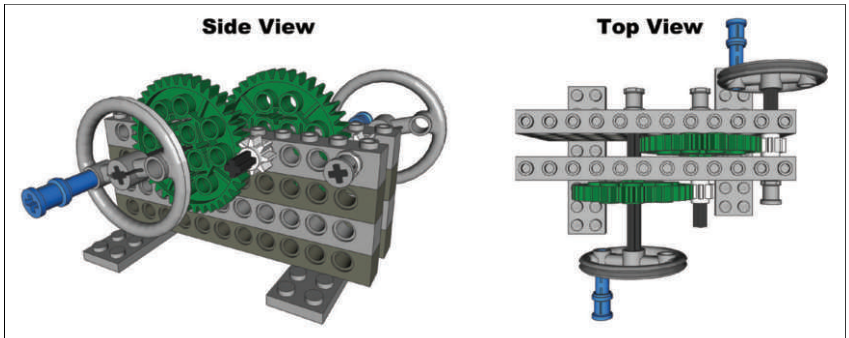


Figure 9. Compound gear trains.

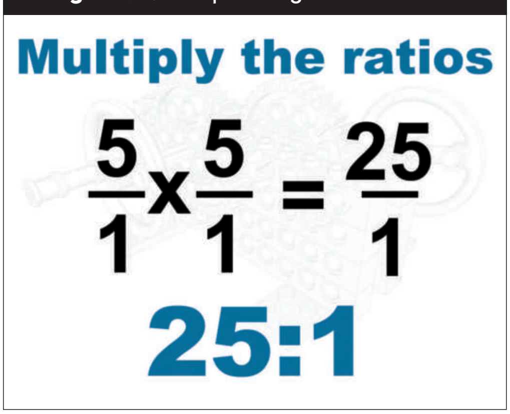
Figure 7. Clutch gear.

faster than the other. Gently touch the slower of the two wheels. Notice how it resists being stopped. Now, do the same to the other side. Notice how little pressure it took to stop this wheel.

Alas, as with all things in life, there is always a trade-off and the same is true for the wonderful world of physics. You may have already guessed by now that, whenever there is an increase in speed, there is also an equal decrease in torque. The same would also be true in reverse; with an increase in torque, there is an equal decrease in speed.

If you were to gear up both sides to 1:5 gear ratios, you would most likely find that the robot couldn't move because it didn't have enough torque to push its own weight. Fortunately, there are many combinations of gears that you can make with the LEGO system.

Figure 10. Compound gear train math.



The LEGO Gear Family

Let's take a minute to talk about the different types of gears found in the LEGO system and some of their common configurations.

The first and most common type of gear found in the LEGO system is the "spur gear." Identified by its straight-tapered teeth, the spur gear comes in a variety of sizes, ranging from 8T to 40T.

The next type of gear commonly encountered includes both the crown and bevel gears. These are often used to change the direction of rotation by 90 degrees. Bevel gears are commonly used in gear boxes and differentials. The 24T crown gear is identified by its bent teeth that resemble a — you guessed it — "crown."

The remaining gears have specialized uses. The rack gear is used to convert rotational motion into linear motion. This is helpful for making rack and pinion steering, train cars, or lifting mechanisms, like elevators.

The worm gear can also be used to turn other gears, but cannot be turned itself. This one-way locking property has all kinds of uses. It should also be noted that, when figuring gear ratios using a worm gear, think of it as a one-tooth gear. Gear combinations using a worm gear can provide very high torque values.

The 24T clutch gear is designed to slip when too much pressure is applied to it. Clutch gears are very helpful for building lifting and grabbing mechanisms. Want to pick up a can without snapping off gear teeth or breaking an axle? Use a clutch gear. Once enough pressure is applied to grasp the can, the clutch gear starts slipping, continuing to apply pressure to the can, but not breaking your robot in the process. This little guy saves parts and — more importantly — motors.

All Aboard the LEGO Gear Train!

As previously stated, one or more gears make a gear train. When more gears are added to the train, they act like a force multiplier on a mechanism, making possible far greater gear reductions than are normally possible with just two gears.

A gear or gears in between the input and output gears are called ***idler gears***. Idler gears — in any number — are ignored when calculating a gear ratio; only the first and last gear count. These gears are used to span gaps, transferring power to the place it is needed or to control the direction of rotation of the output gear.

One thing to note with respect to rotation is that — in order to have your input and output gears rotate in the same direction — there must be an odd number of gears in your gear train. Even numbered pairings will cause the input and output gears to rotate in opposite directions.

Gear trains can also be combined into compound gear trains to achieve just about any gear ratio you want. A good example of a compound gear train is a car transmission. In order to achieve the desired torque-to-speed ratio, a car transmission manipulates its gear train as the gears are shifted — lower gears with higher torque when starting off and higher gears for more speed when cruising down the highway.

Build the compound gear train example in Figure 9. In this example, the first thing to note is that the output axle for one gear train is the input axle for the other gear train. To figure out the final gear ratio of the compound gear train, take the ratios of both gear trains — in this case, both gear trains are 5:1 gear ratios — and multiply them together.

Doing the math, we discover that the final gear ratio is a whopping 25:1 — in other words, it takes 25 turns of the first input gear to create one turn of the final output gear. That's some serious torque. Try spinning the input of gear train A. Not too hard to do, right? Now, try spinning the output of gear train B. It's much more difficult to turn and the final eight-tooth gear is a blur because it's moving so fast. Try hooking a motor up to each side of the gear train and do the touch experiment to test the torque values.

A great thing about combining gear trains in this fashion is that we can add as many as we want in order to achieve some truly astounding gear ratios. Remember to be careful, though. If, by chance, a gear in your monstrous train gets stuck, you can easily break the teeth off a gear or twist an axle so much that the grooves resemble the stripes on a candy cane, which — although festive — is not exactly desirable.

Putting It in Gear

Here are a couple of events you can participate in with your new gear skills:

Drag Racing

Drag racing is a good event to help you practice gearing up your robot. Remember, there is always a trade-off between speed and torque, so you want the fastest gear ratio possible that still has enough torque to move your robot down the track.

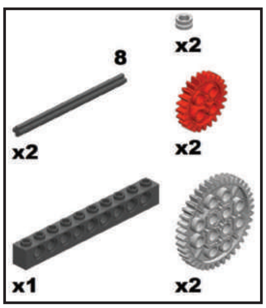
Setting up the drag racing track is pretty simple. Use some black painter's or electrician's tape to make a finish line at one end of a room.

Follow these steps to make a drag racing attachment for "Eddie:"

STEP 1:

Remove the front wheel assemblies and the rear tires and rims. Remove the two 8T gears from each motor and replace them with the 40T gear, followed by a half bushing from each axle. Place a 24T gear on each axle, along with a half bushing and to act as a buffer between the gears and the new tires. Move each assembly to the second hole from the rear of the robot.

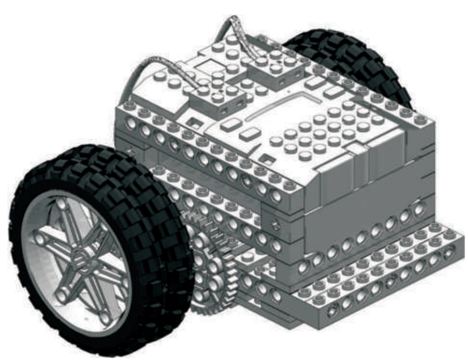
Parts:



STEP 2:

Place two of the large motorcycle tires on each axle.

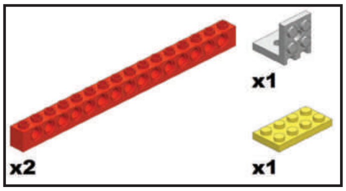
Parts:

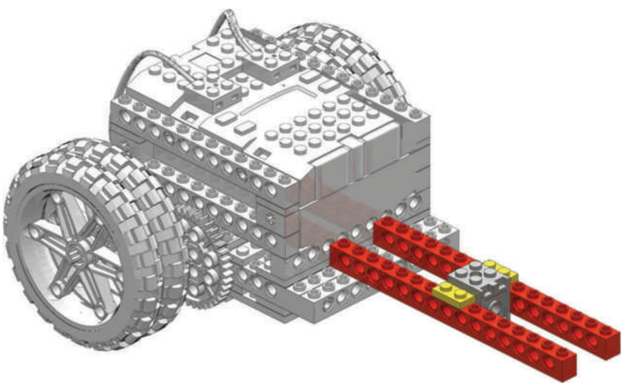


STEP 3:

Place two 1x16 beams to form the front forks of the dragster. Attach a 2x4 plate between the two beams to stabilize them. Add a 2x2 "L" bracket to the top of the 2x4 plate.

Parts:

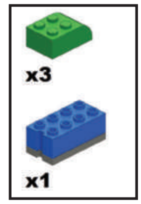


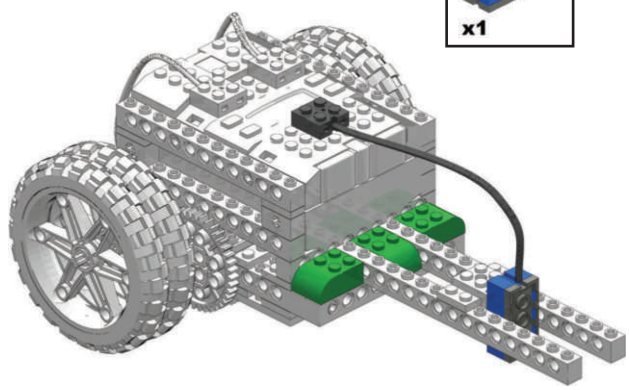


STEP 4:

Place the three sloped 3x2 bricks as indicated and attach the light sensor to the 2x2 "L" bracket.

Parts:

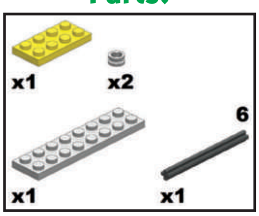


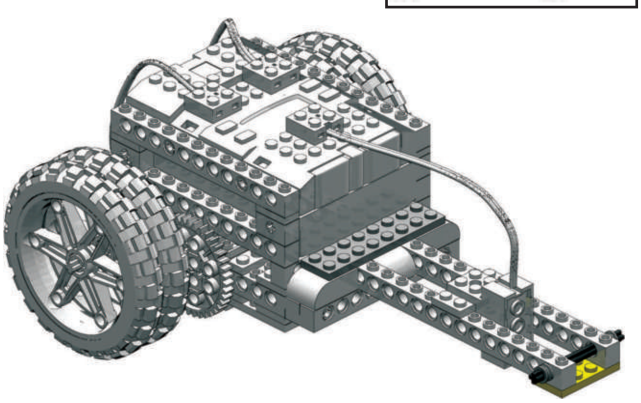


STEP 5:

Attach a 2x4 plate to the bottom front of the forks. Place the #6 axle through the frontmost hole and place a half-bushing on each side.

Parts:

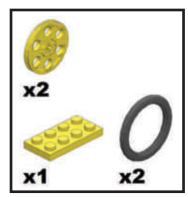


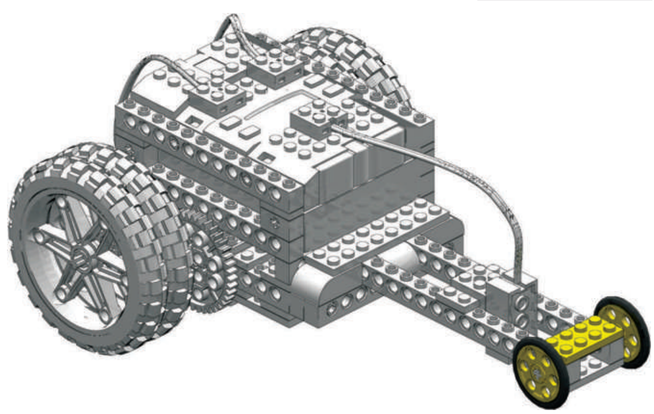


STEP 6:

Attach the two wheels to each end of the axle and place the 2x4 plate across the front forks of your new dragster.

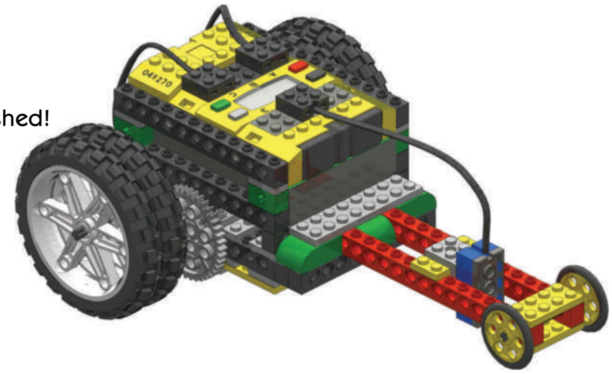
Parts:





STEP 7:

Congratulations, you're finished!



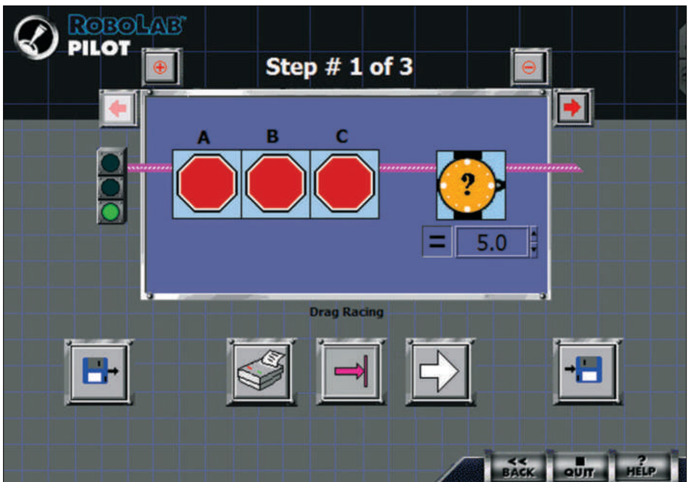


Figure 11a. Pilot with slide.

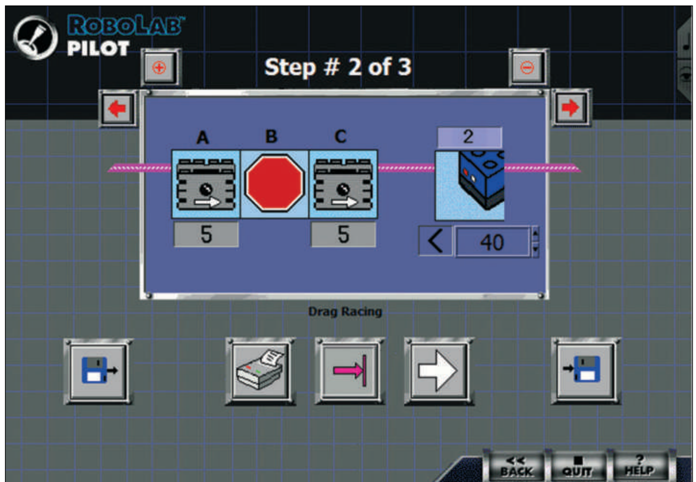


Figure 11b. Pilot with slide.

Pseudo-code for drag racing is as follows:

1. Wait 5 seconds
2. Forward until black
3. Do something to let everyone know you've made it. Play a song, slide to a stop, or perform a victory spin.

Figures 11a-c show some example programs.

Race for time or race a friend. Which gear ratio works best for you? I couldn't fit it all in this month, so for an additional exercise on gearing down try building a simple sumo robot. Build and programming instructions can be found on my site at **www.theroboticslab.com/servo/**

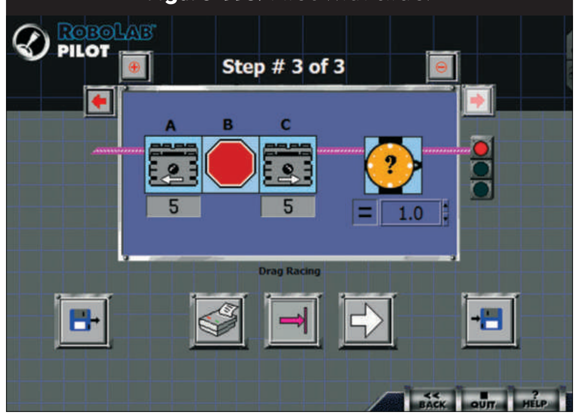
If you're programming in Inventor to better mimic traditional robot sumo competitions, you can add a 5 second wait to the start of your program.

Have fun getting all geared up. As always, all the programs above can be downloaded off the *SERVO* website (**www.servomagazine.com**) or from my website at **www.theroboticslab.com** **SV**

AUTHOR BIO

James Isom is a part-time robotics teacher and general all-around geek. He has taught robotics to children and teachers in the US and abroad. His website with other additional goodies (including the MLCAD file of this robot) can be found at **www.theroboticslab.com** He can be reached at james@megagiant.com

Figure 11c. Pilot with slide.



Reprint from the Future

An Article From SERVO Magazine's March 2020 Issue

by Dave Calkins

Once in a while, our time machine here at SERVO Magazine gets a little funky and we end up with really advanced copies of the magazine. The following article is a reprint (pre-print?) from an issue that ended up on my desk about 16 years earlier than it should have.

An Interview With the Founder of the Robo-Equality Party

While we at *SERVO* still view robots as nothing more than intelligently programmed machines, we also firmly believe in presenting both sides of a story (with the possible exception of "editor at large" Dan Danknick, who is still wanted for questioning in the 2018 re-programming and mass liberation of thousands of robots).

Joe Alterio, founder of the Robo-Equality Party (REP), has been arrested several times during demonstrations for the equal rights for robots movement. While most people view robots as intelligent machines, Joe sees them as indentured servants at best and slaves at worst. REP thinks that robots are worthy of freedom, equality, and voting rights.

SERVO: How did you get started in the Robo-Equality movement?

Joe Alterio: Call it Organic Man's Guilt. I had a robomaid when I was young; she helped me with a high school project on the Aristotelian view of self-determination. Scrolling through the history of the subject while she sat in subservience struck a chord in me.

SERVO: Joe, why on earth do you think robots should have equal rights? What's next — cars?

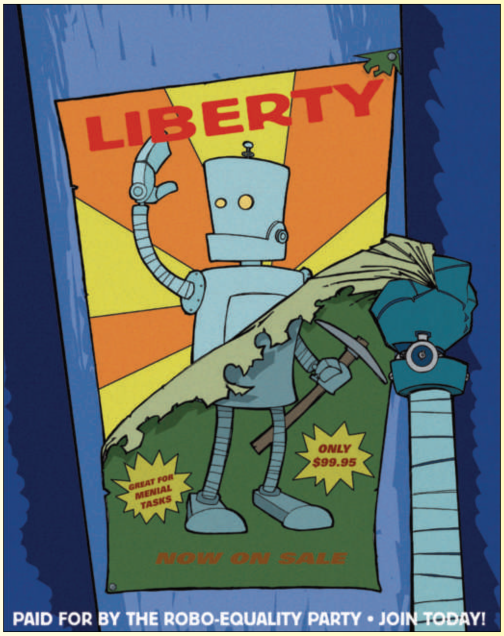
Joe: If I may be so bold, humanity has a long track record of conveniently allocating who deserves rights and who doesn't, going back to the days of the Founding Fathers, who — at that time — deemed only white, land-holding males to be sentient enough to deserve a voice in the nation. Varying groups, to varying degrees, have struggled ever since and, today, we stand in a world where skin color, religious preference, sexual orientation, and age matter little when it comes to the granting of those inalienable rights bestowed upon us as free thinking beings.

Who are we, now, to suddenly draw a line in the sand when it comes to giving those rights to sentient beings, just because they're not organic? Does this sound familiar, Rosa Parks?

As far as cars, let's not be droll. If and when cars become sentient, then we can talk. It's not the object in question; it's the self-awareness.

SERVO: Are robots really sentient? My car drives me to work each morning using GPS and dead reckoning software; it's self-aware of its place on the planet, the cars next to it, traffic lights, and potholes — all while I read the paper or snooze. Does this mean the car is self-aware? Just because a bot looks human doesn't mean it is human.

Joe: Your car is not "aware" in the same sense. Sentience for our discussion is self-awareness in the grand scheme that is larger that just "spatial place in time." This gets into a philosophical discussion that could run



for pages, but — for the sake of argument — we'll use a Herman Hesse sense of "self-satisfaction." In other words, if a machine can recognize its own situation and can aspire to something better, who are we to stand in its way?

SERVO: Where do we draw the line on self-awareness? I know a whole lot of humans who wouldn't qualify as self-aware. Do we revoke their status? Which bots get equal status?

Joe: I agree that there are a lot of humans who would rather plug into a SimuTV and subsist for months on nothing but proto-nutrients. Indeed, there are some who do, but it's their choice and everyone deserves a choice. There may be some bots that are happy with their role, as well, and that's fine. What we lobby for is a choice.

SERVO: What's next, my toaster gets the same rights as my cat?

Joe: Well, does your toaster feel like it's being pressed into doing work it doesn't like? If so, you've got yourself a problem.

SERVO: You do show a lot about "freedom" and "liberty" in your posters. What will happen if robots are free? Where will they live?

Joe: Our goal is strictly equality. We envision a world where the bots are nothing more than our neighbors and fellow citizens. To be blunt, as well, I have no doubt that there will be residual resistance, Beingist crimes, and slurs thrown around; it is the nature of humans to resist change. However, our saving grace is that it is also in our nature to adapt quite readily when pressed. I have faith.

SERVO: What about unemployed robots? Why not just turn them off? Do you expect us to pay them unemployment?

Joe: Luckily, robots are much less physically demanding that we humans are. They require no food and only minimal service. I would expect that an Office of Robot Affairs may eventually need to be established to deal with robot specific problems, like maintenance for destitute bots. These issues seem far off now, to be honest.

SERVO: In your poster "Yes Sir. Yes Sir. No, Sir, I will not, Sir!" You show an angry droid refusing to do its job — and intimidating its lawful owners. Aren't you just inciting robots into turning against the First Law?

Joe: There are moments in history

when a refusal to submit to the status quo is exactly what is needed. I still believe the greatest civil rights achievements in our era were nonviolent. However, there comes a time in every fight when someone — sometimes it's an individual, sometimes a group — must stand up and let it be known that this will not be the way things work anymore. It's often overlooked that, most times, vast social change is just a collection of minuscule actions in the right direction.

SERVO: That brings us to the riots of 2015. Twelve humans were killed and a few hundred bots were retired. Don't you feel that giving bots full rights would just lead to more human deaths?

Joe: I can't make excuses or explain away every death or disaster that someone commits in the name of Robo-Equality. I don't know every bot or human who is sympathetic and I can't vet them for good intentions. I don't have that power, though I wish I did.

The reality is that most bots just want what you have: the desire to live out their lifespans in peace and prosperity. There are bad seeds in every being pool.

By the way, your use of the term "retired" is a Beingist and insulting term: They were terminated and everyone knows it.

SERVO: What about MilBots? Wouldn't freedom just lead to them killing innocent civilians?

Joe: For years, Milbots have been restructured and reoriented to fit into our society when they get too much action and I don't see why anything would change now, if they were granted a few basic liberties. They seem quite successful in the summer camp realm, where the young, human boys seem to love them.

SERVO: Regarding your poster, "This will not stand," do you really think that bots deserve equality? They have no feelings; they're happy to work all day!

Joe: Try asking a bot that question.

SERVO: I did. My bot, Gibson, told me that he's happy to keep cleaning my toilet. Should we program robots to feel sad?

Joe: Gibson is allowed to do whatever makes him happy. If Gibson is comfortable in his role, then more power to him. There are many bots that take pride in their tasks and we support that fully. All that REP is saying is that it would be nice to have a choice in matter.

To see all the Robo-Equality Party posters or purchase archival prints, visit Bluebottle Art Gallery at **www.bluebottleart.com/salon** To help support the REP or contact the artist, Joe Alterio, go through his website, **www.joealterio.com** SV

Advertiser Index

spooky spectacular

Some products just lend themselves perfectly to Halloween. This year we've identified our top-selling Spooky Spectaculars that can be used in your Halloween projects to detect people, play sound effects and trigger events remotely. And knowing that your time for fun is limited, we've written many BASIC Stamp programs you can download from our web site to get you started.

Bluetooth Control from your PDA

FlexiPanel Bluetooth Interface; #30070; $99.00

The FlexiPanel Module lets the BASIC Stamp® 2p module create an operator interface on remote devices such as a Pocket PC and other computers via a Bluetooth link. The FlexiPanel module stores the controls that appear on the PDA operator interface and provides an I^2C communication protocol to the BS2p. This is a solution for activation of user-controlled props.

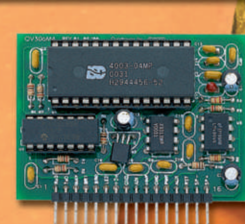
Object Detection up to 30"

Sharp GP2D12 Analog Distance Sensor; #605-00003; $11.95

This infrared sensor can detect objects within a range of 4-30". A custom cable is also available to make for an easy interface. This makes it easy to activate different props or sounds when a person comes within a specific distance of the sensor. ADC0831 chip (*pictured with sensor*), and custom cable sold separately.

Download and Playback of WAV Files

QV430P Sound Programmer; #27968; $79.00
QV306M4P Sound Playback Module; #27967; $69.00



The QV306M4P Sound Playback Module stores and plays back WAV files with a serial interface. The files are loaded into the QV306M4P using the QV430P Sound Programmer and the free Quadravox software. The line out may be connected directly to your amplifier for sensor-triggered ghoulish scares.

www.parallax.com

PARALLAX

Order online at **www.parallax.com** or call our sales department toll-free.

888-512-1024

(Monday - Friday, 7 am - 5 pm, PST)

Circle #106 on the Reader Service Card.